TRABAJO DE GRADO PARA OPTAR AL TÍTULO DE MAESTRÍA EN INGENIERÍA

Estandarización de datos y priorización de algoritmos para un sistema de recomendación

Autor:

Dider León González Arroyave

Asesor:

Edwin Nelson Montoya Múnera

Departamento de Informática y Sistemas

Escuela de Ingeniería

Universidad EAFIT

2019

Agradecimientos

A mis padres y mi hermana por su paciencia, comprensión y apoyo total. Por creer y luchar a mi lado durante este proceso de formación.

Agradezco a mi asesor, Edwin Montoya quien siempre encontró un momento para compartir su conocimiento, evaluar mis avances y proponer mejoras importantes sobre este trabajo, además de promover mi formación mediante la invitación a cursos y eventos alineados a mi perfil e intereses.

Agradezco a la profesora Marta Silvia Tabares por ser una guía muy importante en este proceso, aportando su conocimiento y experiencia.

Agradezco a el profesor David Velásquez por su constante guía, amistad y apoyo. Por compartir su experiencia y motivarme a emplear las mejores practicas en la realización de este trabajo.

Agradezco a Héctor Pico, Edwin Montoya y Ricardo Villegas por su confianza al abrirme las puertas a esta oportunidad de formación.

Agradezco a la universidad EAFIT y sus profesores por contribuir en mi formación profesional y personal.

TABLA DE CONTENIDO

RESUMEN

Un sistema de recomendación es un conjunto de herramientas y técnicas de software que provee sugerencias al usuario de un sistema sobre cuáles ítems le pueden ser de utilidad [1], [2].

Un negocio físico ofrece al consumidor productos o servicios populares, ya que existen limitaciones de espacio que no le permiten presentar muchas opciones en un solo lugar. Por otro lado, los negocios en línea pueden poner a disposición del consumidor cualquier cosa que exista, incluyendo artículos populares y poco populares. Este fenómeno, llamado The Long Tail, obliga a estos negocios a recomendar artículos personalizados a cada usuario [3] logrando así una mayor cobertura del catálogo y una mejor experiencia a sus clientes.

La implementación y despliegue de un sistema de recomendación puede implicar una inversión sustancial, es por esto que muchas empresas de comercio electrónico prefieren externalizar los servicios de recomendación [4]. Este tipo de implementaciones son llamadas recomendaciones como servicio. El desarrollo de estos servicios requiere definir aspectos como la captura y actualización de los datos de entrenamiento, la evaluación, selección e hibridación de algoritmos según los datos disponibles y las interfaces para consumir el servicio [5].

Cada cliente, que requiere un sistema de recomendación, tiene un conjunto de datos con características específicas. Diseñar una solución para diferentes conjuntos de datos implica un proceso de preparación y adaptación a los algoritmos que se van a utilizar, esta etapa representa un esfuerzo significativo.

En la literatura se encuentra una gran variedad de algoritmos de recomendación, no todos presentan los mismos resultados para diferentes conjuntos de datos. Es una

tarea importante seleccionar el algoritmo que mejores resultados ofrece según las características de los datos del cliente.

Este trabajo propone un sistema de recomendación que estandariza los conjuntos de datos y presenta un mecanismo que explora diferentes algoritmos, prioriza y selecciona el que mejor lo hace según un conjunto de métricas definidas.

Se implementa un prototipo del sistema propuesto y se prueba con tres conjuntos de datos diferentes. Se comprueba la estandarización de los datos de entrada para los formatos propuestos, permitiendo el uso de diferentes fuentes de datos. También se usa el prototipo para entrenar, hibridar, evaluar y priorizar los algoritmos en cada conjunto de datos.

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

DICCIONARIO DE TÉRMINOS

Bundle: Grupo de productos se ofrecen a un usuario que ha mostrado interés en alguno de los productos del grupo.

Rating (o calificación): Cualquier tipo de respuesta de un usuario frente a un item, ya sea una calificación explicita o una interacción (implícita).

Review: Calificación textual que da un usuario a un producto o servicio.

Cold-start: Cuando un usuario no ha calificado suficientes items como para calcular la similitud con otros usuarios.

DSA: Data Staging Area – Área de preparación de datos

Ítem: Se usa para referirse a un producto o servicio del que dispone un negocio.

Tags: Son categorías que asigna el usuario a los items presentados por el sistema, representan lo que piensa el usuario sobre ese item.

E-commerce: o comercio electronico, define cualquier negocio o transacción comercial, que implica la transferencia de información a traves de Internet.

Log: Es un registro que se almacena en un archivo o base de datos y que corresponde un evento o acción realizada en un sistema de información.

Feedback: retroalimentación.

Cluster: agrupación.

Ranking: Relación entre un conjunto de elementos, tales que, para uno o varios criterios, el primero presenta un valor superior al segundo y asi sucesivamente.

Throughput: La tasa de transferencia efectiva es el volumen de trabajo o de información neto que fluye a través de un sistema.

Significancia estadística: Es un concepto estadístico asociado a la verificación de una hipótesis. Un resultado se considera estadísticamente significativo cuando es poco probable que haya sido debido al azar.

CAPÍTULO 1: INTRODUCCIÓN

La necesidad humana se describe como la sensación de carencia de algo, unida al deseo de satisfacerla. Clasificar las necesidades es un intento imposible, porque el hombre es capaz de necesitarlo todo, incluso lo que no existe más que en su imaginación. [6]

Cada día aparecen nuevos productos y servicios, estos buscan solucionar una necesidad, manifestada o no, por las personas de la población a la que va dirigida dicha solución. El éxito o fracaso de estos productos y servicios se ve determinado por el uso de estrategias de marketing y publicidad, las cuales buscan atrapar al público objetivo, creando o invocando necesidades ya sean biológicas (primarias) o culturales (secundarias) y aumentando su fidelización y deseo por consumir.

La masificación de Internet en dispositivos como computadores y teléfonos móviles, ha creado un entorno en línea que permite consultar y consumir cualquier producto o servicio, en cualquier momento y desde cualquier ubicación.[7]

La cantidad de opciones disponibles en la web, en combinación con su naturaleza dinámica y heterogénea, ha incrementado la dificultad para encontrar productos o servicios que cumplan los requerimientos e intereses del usuario [8]. En consecuencia, se ha incrementado el interés en sistemas de recomendación que de forma personalizada faciliten la tarea de encontrar información relevante y tomar decisiones.

Un sistema de recomendación se define como un conjunto de herramientas y técnicas de software que provee sugerencias al usuario sobre cuáles ítems le pueden ser de utilidad [1], [2].

Los sistemas de recomendación más populares incluyen métodos basados en contenido, filtros colaborativos, análisis de redes sociales y la combinación entre diferentes técnicas: sistemas híbridos.

Este trabajo se centra en el método Filtro colaborativo, es el más utilizado y se basa en la similitud entre usuarios. Este método asigna cada usuario a un grupo de usuarios con gustos similares y le recomienda artículos que hayan sido de interes para los demás usuarios del grupo.

Los sistemas de recomendación no son un tema nuevo, podrian tener más de 30 años, pero la disposición de grandes cantidades de datos, la constante evolución en la tecnología que los genera y la tendencia a un mundo conectado a internet, propone nuevos retos y necesidades a los negocios e instituciones que quieren encontrar soluciones competitivas en esos datos.

La labor de desarrollar, instalar y configurar este tipo de sistemas puede significar una barrera impasable tanto para desarrolladores experimentados como para usuarios novatos, lo que limita la adopción de los sistemas de recomendación en lugares donde serían muy útiles como pequeños negocios y startups, sitios web de organizaciones y para uso en investigaciones de estudiantes [9]. Por otro lado, la implementación y despliegue de un sistema de recomendación puede implicar una inversión sustancial, es por esto, por lo que muchas empresas de comercio electrónico prefieren externalizar los servicios de recomendación [4] [5]. Este tipo de implementaciones son llamadas recomendaciones como servicio. El desarrollo de estos servicios requiere definir aspectos como la captura y actualización de los datos de entrenamiento, la evaluación, selección e hibridación de algoritmos según los datos disponibles, al igual que las interfaces para consumir los servicios.

A nivel local, tanto en la academia como en la industria, es cada vez más común la disposición de muchos datos y a su vez, la necesidad de implementar sistemas que hagan uso de estos para obtener información relevante. No es óptimo pensar en soluciones a la medida para cada necesidad. Por el contrario, desarrollar un sistema

de recomendación que de forma general permita atender los requerimientos de recomendaciones para diferentes clientes es uno de los objetivos principales de este trabajo.

La integración de fuentes de datos heterogeneas es uno de los retos al desarrollar un sistema de recomendación [16]. Cada negocio genera diferentes tipos de información, y los almacena en diferentes formatos. Todo sistema necesita la información de los usuarios para producir los resultados para los que esta diseñado. Es por esto que un sistema de recomendación, al igual que todo sistema de información, tiene como tarea fundamental la obtención de los diferentes tipos de datos producidos por cada cliente y su posterior transformación que los prepare para ser consumidos por el resto del sistema.

Son procesos comunes en un sistema de recomendación el entrenar los algoritmos a usar, hibridar algoritmos para encontrar mejores resultados, evaluar los algortimos para determinar que tan bien lo hacen y presentar una capa para consumir las recomendaciones y el sistema como tal. Al diseñar un sistema que pueda atender varios clientes en diferentes dominios, es necesario entender que un solo algoritmo no presentará resultados con la misma calidad para todos los clientes. Por lo tanto, aun cuando el algoritmo lo haga muy bien en un escenario, puede no hacerlo tan bien en otro. Burke [17] indica que todo estudio esta por naturaleza limitado por las peculiaridades de los datos y el dominio de recomendación. Por lo anterior, es necesario disponer de varios algoritmos y probar cual lo hace mejor para cada cliente específico.

Este trabajo busca proponer el diseño para un sistema de recomendación que permita estandarizar los conjuntos de datos de entrada, permitiendo proveer un servicio a diferentes clientes en diferentes dominios. Por otro lado, este diseño debe apoyar el proceso de selección del algoritmo óptimo, el cual varia de acuerdo a las caracteristicas específicas de cada conjunto de datos y a los resultados de evaluación en una o varias métricas.

La validación de la propuesta se realizará mediante la implementación de un prototipo que será probado con tres conjuntos de datos, con propiedades y dominios diferentes, con el objetivo de demostrar el uso de la solución en diferentes conjuntos de datos.

Este trabajo está organizado en 5 capítulos así:

Capítulo 1. Introducción

Capítulo 2. Marco conceptual y trabajos relacionados: Presenta la definición de los conceptos básicos, tipos de datos, técnicas y métricas de evaluación propias de un sistema de recomendación, además de otros conceptos relacionados con la elaboración de este trabajo. También presenta trabajos relacionados sobre arquitecturas de referencia y selección de algoritmos.

Capítulo 3: Propuesta de diseño de la solución: Se presenta la arquitectura propuesta con su descripción y los detalles de diseño, modelo para la priorización y selección de algoritmos y desarrollo del prototipo.

Capítulo 4: Validación del prototipo: Se presenta una descripción y análisis sobre los conjuntos de datos con los que se probará el prototipo. Se detalla la preparación requerida para cada conjunto de datos y se presentan los resultados obtenidos para cada uno de estos.

Capítulo 5: Conclusiones y trabajos futuros: Presenta las conclusiones del trabajo, limitaciones y discute trabajos futuros.

OBJETIVOS

Objetivo general:

Diseñar un sistema de recomendación que normalice el uso de conjuntos de datos en diferentes formatos y presente un método de priorización y selección de algoritmos de recomendación basado en filtro colaborativo.

Objetivos específicos:

- Realizar una revisión del estado del arte de algoritmos para sistemas de recomendación.
- Normalizar y estandarizar los datos de entrada para los algoritmos de recomendación a seleccionar.
- Definir el flujo de datos y arquitectura del sistema.
- Implementar un prototipo funcional del sistema propuesto que permita la estandarización de los datos, entrenamiento, evaluación, hibridación y priorización de los algoritmos de recomendación a seleccionar
- Uso del prototipo para la demostración y evaluación de la solución propuesta con diferentes conjuntos de datos.
- Definir servicios para el consumo de las funciones ofrecidas por la solución.

ALCANCE

Este trabajo propone el diseño de un sistema de recomendación que presenta un método que estandariza diferentes conjuntos de datos para ser usados por el resto del sistema. Entrena e híbrida un conjunto seleccionado de algoritmos. Evalúa cada algoritmo y cada hibridación. Por último, prioriza y selecciona el algoritmo con mejor resultado de acuerdo con métricas de evaluación definidas.

El diseño del sistema no está pensado para que el algoritmo seleccionado se actualice cada vez que se reciben datos nuevos, el entrenamiento de los algoritmos debe ejecutarse periódicamente para tener en cuenta los nuevos datos recolectados. De igual forma, el proceso de priorización y selección del mejor algoritmo debería ejecutarse cada vez que las características de la población cambian con cierta significancia, esto se propone para investigaciones futuras, buscando que el sistema aprenda y detecte automáticamente cuando priorizar y seleccionar un nuevo algoritmo.

En la evaluación del prototipo solo se tendrán en cuenta fuentes de información explícitas con calificaciones. Es la fuente de información más común, tanto en el estado del arte como en los conjuntos de datos disponibles para investigación. Sin embargo, el diseño de la solución incluye otras fuentes de información como explícito con comentarios, implícito, características de artículos y características de usuarios.

La implementación y evaluación del prototipo solo hace uso de algoritmos de filtro colaborativo, esta técnica es una de las más utilizadas en el estado del arte y en el desarrollo de soluciones de sistemas de recomendación en el mercado; por otro lado, para esta técnica se encuentran más implementaciones libres para la adaptación tecnológica en el prototipo. El diseño del sistema por otro lado está

abierto al uso de otras técnicas de recomendación como filtro por contenido y popularidad.

Para la evaluación de los algoritmos se usan las métricas RMSE (*Root Mean Square Error*), MAE (Mean Absolute Error), precisión (entre todas las recomendaciones presentadas al usuario cuantas son útiles) y *recall* (de todos los ítems que serán útiles para el usuario, cuantos se presentaron en las recomendaciones). También se tiene en cuenta el tiempo de ejecución del algoritmo. El uso de otras métricas de evaluación se propone para trabajos futuros.

El método de extracción de datos solo admite archivos con formato CSV (*Comma-Separated Values*) (o texto con un separador definido) y JSON (*JavaScript Object Notation*). Los archivos con formato JSON deben estar organizados por líneas, no se permite acceder a objetos anidados.

La ejecución y evaluación de los algoritmos se realiza secuencialmente. Se propone en trabajos futuros usar técnicas de procesamiento en paralelo para esta tarea.

El método de extracción de datos funciona con archivos almacenados sobre el sistema de archivos. Se propone en trabajos futuros, contar con información de archivos almacenados en el sistema HDFS (*Hadoop Distributed File System*) o masivos distribuidos.

El prototipo no incluye la posibilidad de recibir retroalimentación explícita del usuario frente a las recomendaciones presentadas. Esto es una forma de retroalimentar el modelo de recomendación y se propone para trabajos futuros.

El módulo de análisis de sentimientos funciona solo con el idioma inglés para este prototipo, debido al uso de conjuntos de datos en ese idioma. Adaptar otros idiomas se propone para trabajos futuros.

METODOLOGÍA

Design Science Research methodology (DSRM) for information systems

La ciencia del diseño es de importancia en disciplinas orientadas a la creación de artefactos exitosos. Diseño es considerado como el acto de crear explícitamente una solución aplicable a un problema. Las disciplinas en ingeniería aceptan el diseño como una metodología de investigación válida y valiosa ya que otorga valor explícito, incrementalmente, a la solución efectiva y aplicable de un problema [18].

La metodología presenta un modelo que consiste en seis actividades, como se muestra en la *Figura 1*.

La metodología está estructurada en un orden secuencial, sin embargo, no se espera que siempre se proceda en ese orden. Se puede iniciar desde cualquiera de las cuatro primeras actividades y continuar el flujo del proceso. Iniciar en la actividad 1 corresponde a una aproximación centrada en el problema.

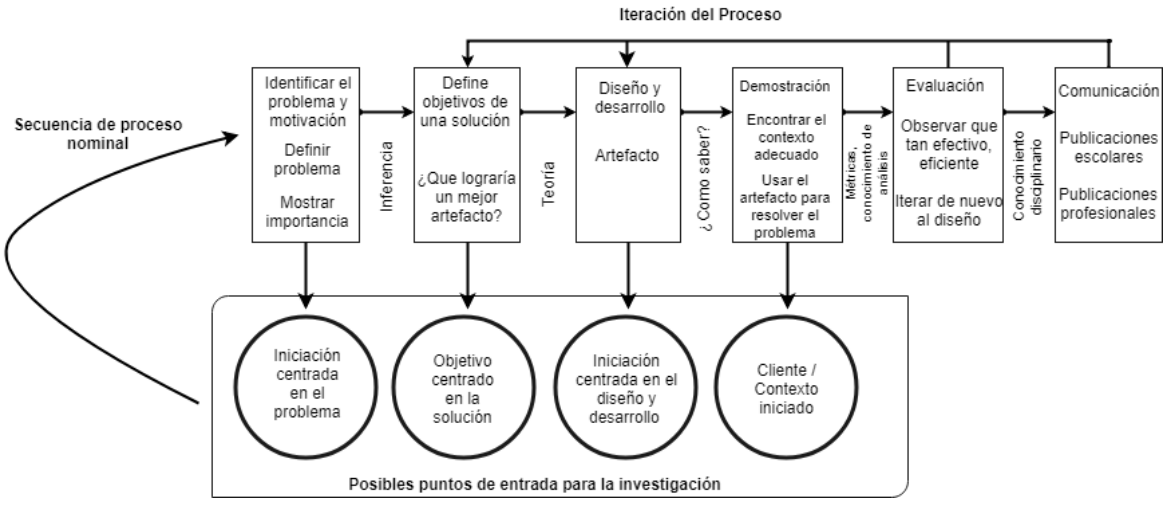

Figura 1: Modelo de procesos propuesto por la metodología DSRM. Tomado de [5]

Actividad 1: Identificación del problema y motivación

En esta etapa se define el problema de investigación específico y se justifica el valor de una solución. La definición del problema permite dividirse conceptualmente en partes más pequeñas para que la solución pueda capturar sus complejidades. La justificación del problema motiva a buscar una solución, entender y aceptar los razonamientos aplicados en la investigación.

Luego de identificar la necesidad de sistemas de recomendación en diferentes proyectos laborales y educativos en los que se participa, surge la motivación por conocer cómo funcionan estos sistemas. Para esto se realizan cursos en línea de sistemas de recomendación y aprendizaje de máquina. Luego de conocer las bases, se hace una construcción del estado del arte, basándose principalmente en los avances presentados por la conferencia ACM RecSys de los años 2016 y 2017, además de los últimos contenidos del tema presentados en revistas indexadas.

Luego de tener un panorama claro de los sistemas de recomendación, se identifican los aspectos más importantes para el diseño de una solución aplicable a diferentes dominios. Se buscan arquitecturas en el estado del arte que permitan guiar el diseño y la construcción del prototipo.

Actividad 2: Definir los objetivos para una solución

Inferir los objetivos para una solución desde la definición del problema y conocer lo que es posible y factible. Pueden ser cuantitativos: como la solución deseada será mejor que las soluciones actuales; o pueden ser cualitativos: Descripción de cómo un nuevo artefacto se espera que provea soluciones a problemas actualmente no resueltos.

Se definen aspectos importantes que determinan el uso de la solución en diferentes dominios, los cuales son: La entrada o extracción de los datos para entrenamiento, la evaluación y selección de algoritmos, un sistema de hibridación que permita

mejorar los resultados de algoritmos individuales y la definición de servicios que permitan consumir el sistema.

Actividad 3: Diseño y desarrollo

Creación del artefacto, este puede ser cualquier objeto en el que se encuentre embebida una contribución de investigación en su diseño. Esta actividad incluye determinar las funcionalidades deseadas y la arquitectura, para luego crear el artefacto.

Esta etapa consiste en pasar de los objetivos al diseño y desarrollo. Requiere incluir conocimientos de la teoría que puedan ser usados en la solución.

De acuerdo a las previas revisiones del marco conceptual y el estado del arte se define una arquitectura de referencia que servirá de guía para el desarrollo de la solución.

Se diseña la arquitectura de la solución, se definen las relaciones y descripción de cada componente. Se diseña el flujo de datos de cada componente de la arquitectura. Se define el mecanismo de estandarización de los datos de entrada y el modelo de priorización de los algoritmos.

Se crea un proyecto en Python y se comienza el desarrollo de cada funcionalidad definida en el diseño de la solución. Primero se abarca la estandarización de los datos de entrenamiento, luego se definen e implementan los algoritmos a utilizar y se adaptan para consumir los datos en el formato resultante de la extracción. Se define e implementa el método de separación del conjunto de datos en entrenamiento y prueba, la función que halla las diferentes métricas de evaluación y almacena los resultados de cada algoritmo, una función de hibridación de algoritmos y una función que aplica el modelo de priorización para la selección de los mejores algoritmos o hibridaciones. Finalmente se definen los servicios para consumir las recomendaciones de los algoritmos previamente seleccionados.

Actividad 4: Demostración

Demostrar el uso del artefacto para resolver una o más instancias del problema. Esto puede involucrar el uso de experimentación, simulación, caso de estudio, prueba u otras actividades apropiadas.

Esta etapa requiere un conocimiento efectivo de cómo usar el artefacto para resolver el problema.

Se definen tres casos de estudio que corresponden a conjuntos de datos en dominios diferentes, se describen estadísticamente y se usan con el prototipo para entrenar, hibridar, evaluar y seleccionar el mejor algoritmo en cada conjunto de datos.

Actividad 5: Evaluación

Observar y medir que tan bien el artefacto soporta la solución del problema. Requiere el conocimiento de métricas relevantes y técnicas de análisis. Dependiendo de la naturaleza del problema y del artefacto, la evaluación puede tomar varias formas: Comparación de la funcionalidad del artefacto con los objetivos de solución anteriormente planteados, medidas cuantitativas de desempeño, resultados de entrevistas de satisfacción, retroalimentación de clientes o simulaciones.

Al final de esta actividad se debe decidir si iterar de vuelta a la actividad 3 para tratar de mejorar la efectividad del artefacto o continuar a la siguiente actividad.

El prototipo será evaluado en su capacidad para usar diferentes conjuntos de datos y seleccionar el algoritmo óptimo según el modelo definido. por lo tanto, debe extraer de forma correcta los datos en los tres casos de estudio. Usar los datos para entrenar, hibridar, evaluar y seleccionar los mejores algoritmos.

Se realiza una prueba estadística en algunos casos de estudio para demostrar que los algoritmos seleccionados no son escogidos al azar y que tienen una probabilidad alta de ser los mismos para cualquier muestra de la población.

Actividad 6: Comunicación

Comunicar a investigadores u otras audiencias relevantes, el problema y su importancia, el artefacto, su utilidad y novedad, el rigor de su diseño y su efectividad.

La comunicación de este trabajo se presenta por medio de este documento y la disertación de los resultados.

CAPÍTULO 2: MARCO CONCEPTUAL Y TRABAJOS RELACIONADOS

SISTEMAS DE RECOMENDACIÓN

Las recomendaciones corresponden a un proceso social y natural en el que se requiere el apoyo de otras fuentes para tomar decisiones sobre alternativas que no se conocen bien [19]. Cuando se pide una sugerencia de otra persona, cuando se lee el análisis de un libro o película, cuando se encuentran ciertos productos en los puntos de pago de un supermercado; estos son algunos ejemplos de recomendaciones.

A medida que los sitios web de e-commerce comenzaron a desarrollarse, los usuarios encontraban muy difícil llegar a los productos o servicios más adecuados, entre una inmensa cantidad de ítems ofrecidos por estos sitios.

Los sistemas de recomendación son un tipo especial entre los sistemas de filtrado de información [20]. Proveen un conjunto de técnicas y herramientas de software que buscan apoyar el proceso de toma de decisiones en los usuarios de un sistema, proporcionando sugerencias de los artículos que pueden ser de utilidad para el usuario objetivo.[17][21]

Un "**ítem**" o artículo, es el término generalmente usado para denotar lo que el sistema le recomienda a los usuarios. Un sistema de recomendación normalmente se enfoca en un tipo específico de artículo y su diseño, interfaz gráfica y técnica principal de recomendación buscan proveer sugerencias útiles y efectivas para ese tipo de artículo.[2]

Las recomendaciones pueden ser **personalizadas**, el sistema identifica quien es el usuario, su perfil, intereses, interacciones con el sistema, entre otros. Y de acuerdo con esa información es capaz de generar diversas sugerencias a diferentes

usuarios. En su forma más simple, las recomendaciones personalizadas se presentan como listas clasificadas de artículos.

También existen recomendaciones **no personalizadas**, estas son más sencillas de generar [2] y recomiendan lo que es popular y relevante para todos los usuarios, esto puede ser una lista con los n artículos principales o más populares del sitio; El sistema no identifica quien es el usuario o tiene muy poca información sobre éste como para entregarle una recomendación personalizada.

Para generar recomendaciones, el sistema trata de predecir cuales son los productos o servicios más apropiados para el usuario, basándose en sus preferencias o restricciones. Para completar esta tarea, el sistema debe recolectar las preferencias de sus usuarios, las cuales pueden ser expresadas de forma explícita o implícita [2]. Una preferencia **explícita** se refiere a un valor o asignación otorgado conscientemente por el usuario a un artículo determinado. Una preferencia **implícita** es la obtenida desde los comportamientos del usuario, es información que la persona no está dando conscientemente, por ejemplo, cuando se visita un sitio web, cuando se visualiza un producto, el tiempo que se pasa en un lugar, las veces que se escucha una canción o se ve una película, la compra de un producto, entre otros.

Ricci et al. [2] enuncia algunos objetivos importantes de los sistemas de recomendación separándolos en los dos actores a los que sirve:

1. Los proveedores de productos o servicios, quienes implementan o adaptan el sistema de recomendación a su negocio, buscan obtener ciertos beneficios como son: incrementar el número de artículos vendidos, vender artículos más diversos, incrementar la satisfacción y fidelidad del usuario, entender mejor lo que quieren los usuarios para mejorar así su negocio [22].
2. Los usuarios, que también deben reconocer los beneficios que les presta un sistema de recomendación, buscan que este soporte las tareas y metas del usuario, como lo son: Encontrar algunos artículos buenos, encontrar todos

los artículos requeridos, recomendaciones de acuerdo con el contexto del usuario, recomendaciones como secuencia o conjunto de ítems, posibilidad de explícitamente mejorar el perfil para obtener recomendaciones más acertadas y novedosas, capacidad de expresarse en el sistema y ayudar a otros por medio de calificaciones y comentarios, entre otros.

Datos y fuentes de conocimiento

Para construir recomendaciones, el sistema debe obtener información desde varios tipos de datos, los cuales son principalmente acerca de los artículos que recomienda y los usuarios objetivo del sistema. Estos datos pueden ser muy diversos y depende de la técnica de recomendación el poder usarlos; a esto se le llama nivel de conocimiento, por ejemplo, una técnica pobre en conocimiento es la que solo usa datos muy básicos como las calificaciones de un usuario para los artículos. Una técnica rica en conocimiento es la que usa fuentes de información adicionales como descripciones ontológicas de artículos o usuarios, relaciones sociales de los usuarios, reglas, entre otros.

Una clasificación general de los datos usados por todo sistema de recomendación se refiere a tres tipos de objetos [2]:

1. Ítems o artículos: Son los objetos que se recomiendan y pueden ser categorizados por su complejidad y su valor o utilidad para el usuario. Algunas técnicas de recomendación permiten usar las características o propiedades que describen los artículos y aprender como la utilidad del artículo depende de sus características.
2. Usuarios: Los usuarios del sistema. Pueden tener metas y características muy diversas. Para entender esto, el sistema busca explotar la información disponible acerca de cada usuario. La información a modelar depende de la técnica de recomendación. El modelo del usuario juega un papel central en el sistema y es el que contiene los datos del usuario y representa sus preferencias y necesidades. Un sistema de recomendación puede verse

como una herramienta que genera recomendaciones por medio de la creación y explotación de modelos de usuario.[23]

Un usuario también puede ser descrito por los datos que representan sus patrones de comportamiento en el sistema, las relaciones entre usuarios y el nivel de confianza entre esas relaciones.

3. Transacciones: Interacciones entre el usuario y el sistema, similares a los *logs* de datos que contienen información importante generada durante la interacción humano-computadora; esta información es requerida por el algoritmo de recomendación que esté usando el sistema. Una transacción puede contener la referencia al artículo seleccionado por el usuario y una descripción del contexto. Si está disponible, la transacción también puede incluir una valoración explicíta ingresada por el usuario. Calificación es la forma más popular de transacción de datos usada por los sistemas de recomendación, estas pueden ser explícitas o implícitas, las calificaciones explícitas indican que tan relevante o interesante es un artículo para el usuario [8]. Estos tienen una escala definida y pueden presentarse en una variedad de formas [2][24]:
    - Calificaciones numéricas: representadas en una escala numérica, por ejemplo: de 1 a 5.
    - Calificaciones ordinales: Al usuario se le pide seleccionar el término que mejor describe su opinión, por ejemplo: muy de acuerdo, de acuerdo, neutral, en desacuerdo, muy en desacuerdo.
    - Calificaciones binarias: El usuario solo tiene dos opciones, le gusta o no le gusta.
    - Respuestas unarias: indican que el usuario ha observado o comprado un artículo como una calificación positiva. En este caso la ausencia de una calificación solo indica que no se tiene información que relacione el usuario con el artículo, no se puede considerar la ausencia de calificación como un valor negativo.

- Asociación de etiquetas: El usuario asigna etiquetas o categorías a los artículos presentados por el sistema, estas etiquetas representan que piensa el usuario sobre un artículo. Por ejemplo, asignar a una película las etiquetas: horror, miedo, paranormal.
- Lops et al. [8] se refiere también a comentarios textuales como una forma de calificación explicíta. Los comentarios sobre un artículo son recolectados y presentados a los usuarios para facilitar el proceso de toma de decisiones, por ejemplo, cuando el artículo ha sido apreciado por la comunidad. Los comentarios son útiles, pero pueden sobrecargar al usuario ya que debe leer e interpretar cada comentario para decidir si es positivo o negativo. La literatura propone técnicas avanzadas desde áreas de investigación relacionadas con el análisis de sentimientos en texto [8], que permiten a los sistemas de recomendación analizar automáticamente este tipo de contenidos y usarlos como calificaciones explícitas para generar el modelo de recomendación.

Para transacciones que recopilan calificaciones implícitas, el sistema debe inferir la opinión del usuario basándose en sus comportamientos o acciones. Estos métodos se basan en asignar una puntuación que indica la relevancia de las acciones específicas del usuario hacia un artículo. La ventaja principal de estas transacciones es que no requieren que el usuario se involucre directamente. Sin embargo, pueden existir sesgos que ocasionen que la calificación implícita no represente la realidad de la interacción, por ejemplo, recibir una llamada telefónica mientras se lee. El sistema asumirá que el usuario aun está pasando tiempo en la actividad de lectura [8].

Con el objetivo de crear y actualizar el perfil del usuario, se debe recolectar de algún modo sus reacciones hacia los artículos del sistema y almacenarlas en un repositorio. Esas reacciones son llamadas anotaciones o *feedback* y son usadas junto a otra información, como las descripciones de los artículos, para alimentar el

proceso de aprendizaje de un modelo útil que permita predecir la relevancia de nuevos artículos que serán presentados al usuario. Los usuarios también pueden definir explícitamente sus áreas de interés como un perfil inicial sin proveer ningún *feedback* [8].

Espacio de diseño técnico de un sistema de recomendación

Resnick et al. [19] plantea un espacio de diseño técnico para un sistema de recomendación en el que se consideran 5 dimensiones:

1. *El contenido de una evaluación:* Puede contener desde un valor binario hasta una anotación textual no estructurada. Algunos ejemplos serían: numérico de 1 a 7, segundos, mención de una persona o un documento, entre otros.
2. *Entradas explícitas o implícitas:* Una entrada puede ser explícita o implícita.
3. Las recomendaciones pueden ser *anónimas*, etiquetadas con la *identidad de la fuente* o etiquetadas con un *seudónimo*.
4. *Como agregar las evaluaciones:* Como se trabajarán las diferentes fuentes de evaluación en el sistema, por ejemplo: aplicar ponderaciones a las evaluaciones, combinar evaluaciones con análisis de contenido, entre otros.
5. *Uso de las recomendaciones:* Como se presentan las recomendaciones obtenidas por el sistema, por ejemplo: filtrar, ordenar y mostrar.

Dimensiones del espacio de dominio

Resnick et al. [19] separa el espacio de dominio de un sistema de recomendación en dos: Los tipos de elementos que se recomiendan y las personas entre las que se comparten evaluaciones.

Para los tipos de elementos que se recomiendan, se consideran cuatro dimensiones:

1. *Tipo de elemento:* Persona, URL, artículo, entre otros.

2. *Cuantos:* miles por dia, millones.
3. *Tiempo de vida:* 1-2 semanas, varios años.
4. *Estructura de costo:* determina el impacto o costo que puede tener recomendar un mal artículo, o no recomendar un buen artículo.

Para las personas que proveen y consumen las recomendaciones, se consideran cuatro dimensiones:

1. *Quien provee las recomendaciones:* los usuarios o fuentes desde donde se toma la información para generar recomendaciones. Los que generan datos que alimentan el sistema de recomendación.
2. *Densidad de las recomendaciones:* Como son los conjuntos de recomendaciones según las tendencias de evaluar muchos artículos en común. Densas o escasas.
3. *Consumidores:* a quien están dirigidas las recomendaciones.
4. *Variabilidad en el gusto de los consumidores:* Que tanto varían los gustos de los consumidores frente a los artículos recomendados.

TÉCNICAS NO PERSONALIZADAS DE RECOMENDACIÓN

El sistema no requiere información del usuario al que quiere recomendar artículos, usa otras fuentes de información para generar el listado de artículos a recomendar.

Popularidad de artículos

El modelo de popularidad es un enfoque base que recomienda al usuario los artículos más populares que este aun no ha consumido. Este modelo suele proveer buenas recomendaciones, basándose en “la sabiduría de la mayoría” o en los artículos que la mayoria han consumido y valorado positivamente.

El sistema genera un ranking o clasificación de los artículos según la popularidad de estos. La popularidad se puede calcular basándose en el número de

calificaciones relevantes obtenidas por cada artículo. Por ejemplo, en un sistema con rango de calificaciones de 1 a 5 (siendo 5 la mejor calificación), el artículo que tenga mayor número de calificaciones iguales a 5 obtendra la primera posición en la lista y el artículo con menos calificaciones iguales a 5 obtendrá la última posición [25].

Esta técnica generalmente no presenta al usuario la lista completa de artículos, por el contrario, se seleccionan los *n* primeros artículos de la lista para ser recomendados. La misma lista de recomendaciones se presenta a todos los usuarios, ya que al ser una técnica no personalizada, no se tiene disponible información del usuario actual y por lo tanto no se puede entregar una recomendación personalizada. Esta técnica es útil cuando no se tiene información suficiente para utilizar otras técnicas personalizadas de recomendación, por ejemplo, el caso de usuarios nuevos en el sistema.

TÉCNICAS PERSONALIZADAS DE RECOMENDACIÓN

Con el objetivo de recomendar artículos, el sistema necesita predecir o al menos comparar la utilidad de esos artículos para un usuario y luego decidir cuáles recomendar. Adomavicius et al. [26] modelan el grado de utilidad del usuario *u* para el artículo *i* como una función de valores reales R(*u*,*i*). Para un algoritmo como filtro colaborativo, la tarea fundamental es predecir el valor de R sobre pares de usuarios y artículos, considerandose como $\hat{R}$ la estimación que calcula el sistema de recomendación sobre la función real R.

Para un usuario *u* el sistema debe predecir la utilidad que tienen un conjunto de artículos $\hat{R}(u, i_1), \dots, \hat{R}(u, i_N)$, luego recomendará los artículos $i_{j1}, \dots, i_{jk}$ $(k \leq N)$ con la predicción de utilidad más alta [2].

La predicción de la utilidad es calculada con algoritmos específicos, que según su diseño, usan una o varias fuentes de información sobre el usuario, los artículos y hasta otros usuarios. para determinar si un artículo es de utilidad o no para el usuario objetivo [17].

Las técnicas de recomendación pueden ser distinguidas con base en las fuentes de conocimiento que requieren para producir recomendaciones [17].

Filtro basado en contenido

En un sistema de recomendación basado en contenido se asume la disponibilidad de información valiosa que describa la naturaleza de cada artículo, por ejemplo en la forma de un vector de características x_i. Para cada usuario u un perfil de preferencias como un vector x_u es obtenido desde el contenido de los artículo que han obtenido algun tipo de calificación por ese usuario [24].

El sistema aprende a recomendar artículos que son similares a otros artículos con los que el usuario objetivo ha interactuado en el pasado y le han gustado. La similitud entre los artículos es calculada con base en sus características asociadas [2]. El sistema busca, por medio de técnicas como similitud de coseno [24], emparejar los atributos del perfil del usuario x_u que contienen sus preferencias e intereses, con los atributos de un artículo x_i para luego entregar nuevas e interesantes recomendaciones [8].

El análisis de documentos o descripciones de artículos previamente evaluados por un usuario, permite la construcción de un perfil de intereses del usuario basándose en las características del objeto, obtenidas desde el texto [8] con técnicas como TF-IDF (Term Frequency Inverse Document Frequency) [24].

Este enfoque tiene algunas ventajas frente a otros enfoques como filtro colaborativo, estas ventajas son [8]:

- Independencia de otros usuarios: Para construir el perfil del usuario solo se necesitan las calificaciones que ese mismo usuario ha proporcionado. No requiere información de otros usuarios como en el caso de filtro colaborativo.
- Transparencia: Es fácil explicar al usuario como funciona el sistema de recomendación, listando las características de contenido que ocasionaron que cierto artículo fuera sugerido en la lista de recomendaciones.
- Nuevos artículos: El sistema es capaz de recomendar artículos que aun no han sido evaluados por el usuario.

Este enfoque también tiene varias deficiencias [8]:

- Analisis limitado de contenido: Se requiere de información suficiente que permita al sistema discriminar los artículos que le gustan al usuario de los que no le gustan. Se requiere un conocimiento del dominio y la disponibilidad de características principales que describan los intereses del usuario pueden no ser suficientes para todos los casos.
- Sobre – especialización: El sistema sugiere artículos para los que su puntaje es mayor cuando sus características se igualan con las del perfil del usuario, esto ocasiona que las recomendaciones siempre contengan artículos similares a los que ya ha calificado el usuario. Estas recomendaciones tienen un grado limitado de novedad por lo que es difícil que el usuario encuentre algo inesperado y tenga la oportunidad de conocer nuevos artículos. La diversidad en las recomendaciones mejora la satisfacción del usuario [27] [28]. A este problema también se le llama *serendipia*.
- Nuevos usuarios: Deben recolectarse suficientes calificaciones del usuario antes de que el sistema realmente pueda entender sus preferencias y entregar recomendaciones precisas, esto representa un problema para los nuevos usuarios que ingresan al sistema.

Filtro colaborativo

Filtro colaborativo (o social) [24] es considerado como una técnica de recomendación muy popular y ampliamente implementada [2]. A diferencia de filtro basado en contenido que usa el contenido de los artículos que fueron calificados por el usuario, filtrado colaborativo usa las calificaciones del usuario objetivo además de las calificaciones de todos los demás usuarios del sistema.

La implementación más simple de este enfoque, recomienda al usuario activo artículos que gustaron a otros usuarios que el sistema considera que tienen preferencias similares. La similitud en las preferencias de dos usuarios es calculada con base en la similitud del historico de calificaciones proporcionado por estos usuarios. La idea principal es que la calificación que le puede dar un usuario u a un nuevo artículo i será muy similar a la calificación que le dio otro usuario v, si u y v han calificado otros artículos de forma similar. De igual forma, es muy probable que el usuario u califique muy similar dos artículos i y j, si otros usuarios le han dado calificaciones similares a ese par de artículos [24].

Ventajas de filtro colaborativo frente a filtro basado en contenido:

- Es posible recomendar artículos para los que no hay disponible contenido, o es difícil de obtener, ya que solo se requieren las calificaciones presentadas por otros usuarios.
- Las características de contenido pueden ser un mal indicador de la calidad del artículo para el usuario. Filtro colaborativo basa la calidad de un artículo en las calificaciones entregadas por otros usuarios.
- Filtro colaborativo puede recomendar al usuario artículos con diferentes contenidos ya que los usuarios similares pueden mostrar interés en variedad de artículos. Mayor diversidad en los artículos recomendados.

Los métodos usados por filtrado colaborativo se pueden agrupar en dos clases:

1. Vecindario (*neighborhood*), también llamada basada en memoria (*memory-based*) o basada en heurísticas *(heuristic-based*) [26]: Las calificaciones

usuario – artículo almacenadas en el sistema son usados directamente para predecir calificaciones de nuevos artículos, esto se puede hacer de dos formas :

- Sistemas basados en el usuario: Evalúa el interés de un usuario u hacia un artículo i usando las calificaciones que otros usuarios han dado al artículo i. Estos usuarios son llamados vecinos (neighbors) del usuario u y son aquellos usuarios v cuyas valoraciones para los artículos calificados por ambos usuarios, están más correlacionadas con las de u.
- Sistemas basados en el artículo: Predicen la calificación que le dará un usuario u a un artículo i, basado en las calificaciónes que ha dado u a artículos similares a i. Dos artículos son similares si varios usuarios del sistema han calificado estos artículos de forma similar.

2. Basado en el *modelo* (*model-based*): Se usan las calificaciones para aprender un modelo predictivo. La idea general es modelar las interacciones de usuarios y artículos con factores que representen las características latentes de cada usuario y cada artículo en el sistema. Este modelo se entrena con los datos disponibles y finalmente se usa para predecir las calificaciones que dará el usuario a nuevos artículos.

   El enfoque basado en el *modelo* es superior al enfoque basado en Vecindario, en la tarea de predecir calificaciones [24].

El enfoque basado en Vecindario es superior al enfoque basado en el modelo, en cuanto al nivel de serendipia que entrega en sus recomendaciones. Serendipia extiende el término de novedad y ayuda al usuario a encontrar artículos que de otra forma no habría encontrado [24].

Demográfico

Este tipo de sistema recomienda artículos basándose en el perfil demográfico del usuario. Se asume que diferentes recomendaciones deberían ser generadas para

diferentes nichos demográficos, por ejemplo, recomendar ciertas páginas web a un usuario de acuerdo a su idioma, país o edad.

Basado en el conocimiento

Los sistemas de recomendación tradicionales (filtro basado en contenido y filtro colaborativo) son útiles para recomendar productos con base en su calidad o gusto, como lo son libros, películas o noticias. Sin embargo, en productos como vehículos, apartamentos, computadores o servicios financieros, estos enfoques no son la mejor opción ya que por un lado no son comprados muy frecuentemente, lo que hace inviable el conseguir numerosas calificaciones para un artículo (requerido por filtrado colaborativo). Por el lado de filtro basado en contenido, los usuarios no estarán satisfechos en recibir recomendaciones basadas en preferencias sobre ciertos artículos consumidos mucho tiempo atrás [29].

Estos sistemas recomiendan artículos basándose en un conocimiento específico del dominio sobre como las características del artículo cumplen con las necesidades y preferencias del usuario, y que tan útil es para este. Los sistemas basados en conocimiento pueden funcionar mejor que otros enfoques al inicio de su despliegue, pero si no se equipan con componentes de aprendizaje, pueden ser superados por otros métodos que hagan uso de *logs* de interacciones entre el humano y la computadora [29].

Los sistemas de recomendación basados en el conocimiento pueden ser de dos tipos[2]:

- Basados en casos: Una función de similitud estima que tanto las necesidades del usuario coinciden con las recomendaciones. [30]
- Basados en restricciones: hace uso predominante de las bases de conocimiento que contienen reglas explícitas acerca de cómo se relacionan los requerimientos del usuario con las características de los artículos. [29].

Este tipo de sistemas no sufre del problema de *cold-start* o inicio frio ya que los requerimientos par las recomendaciones son directamente obtenidos en una sesión de expertos del dominio. Sin embargo, requiere un trabajo duro para convertir el conocimiento de expertos del dominio en una representación formal ejecutable por el sistema [29].

Basado en la comunidad

Este tipo de sistema, también llamado sistema de recomendación social [2], recomienda artículos basado en las preferencias de los amigos del usuario. La evidencia muestra que las personas tienden a confiar más en las recomendaciones de sus amigos que en las recomendaciones de individuos similares pero anónimos [31].

Este tipo de sistema de recomendación modela y adquiere información acerca de las relaciones sociales del usuario y las preferencias de sus amigos. Las recomendaciones son basadas en las calificaciones que proveen los amigos del usuario.

Sistemas de recomendación híbridos

Estos sistemas se basan en la combinación de técnicas y buscan que las ventajas de una técnica cubran las desventajas de la otra. Por ejemplo, los métodos basados en filtrado colaborativo no pueden recomendar artículos que no tienen calificaciones por lo tanto hay problemas cuando entra un nuevo artículo al sistema. Esta limitación no ocurre con filtro basado en contenido ya que las predicciones están basadas en la descripción del artículo y no en sus calificaciones. Este problema es llamado cold-start o inicio frio y su solución es uno de los motivos principales para implementar sistemas híbridos, combinando técnicas que usan diferentes fuentes de conocimiento [17].

No existen motivos para no combinar técnicas de recomendación que usen igual fuente de conocimiento, por ejemplo, combinar dos técnicas diferentes de filtro basado en contenido [17].

Burke [17][32] identifica siete tipos diferentes de sistemas híbridos:

1. Ponderado: Se combina numéricamente la puntuación de diferentes componentes de recomendación. Cada componente asigna una puntuación a un artículo dado y luego las puntuaciones son combinadas usando una fórmula lineal. En este trabajo se usara esta técnica de hibridación, empleando como peso de la ponderación el valor total del rendimiento del algoritmo en las diferentes metricas de evaluación definidas.
2. Selección: El sistema selecciona entre diferentes componentes de recomendación y aplica el seleccionado. Para diferentes perfiles de usuario, diferentes recomendadores deberían ser seleccionados. Se asume la disponibilidad de un criterio confiable sobre el cual basar las decisiones de selección de un recomendador u otro.
3. Mezclado: El sistema presenta juntas las recomendaciones generadas por diferentes recomendadores. Comúnmente se realiza la fusión de las recomendaciones basándose en la calificación predecida o en el nivel de confianza para cada recomendador.
4. Combinación de características: Las características derivadas desde diferentes fuentes de conocimiento son combinadas y sirven como entrada para un solo algoritmo de recomendación. La idea es inyectar las características de una fuente (por ejemplo, filtro colaborativo) en un algoritmo diseñado para procesar datos con una fuente diferente (por ejemplo, filtro basado en contenido). Para el anterior ejemplo, las características que normalmente serían procesadas por filtrado colaborativo ahora son usadas como entrada para filtro basado en contenido y procesadas por este último.

5. Aumento de características: Se usa una técnica de recomendación para calcular una o varias características, las cuales serán parte de la entrada de otra técnica de recomendación.
6. Cascada: La idea es crear un híbrido con una jerarquía estricta, en la cual un recomendador débil no pueda anular las decisiones tomadas por un recomendador fuerte, pero puede refinarlas.
7. Meta-nivel: Se aplica una técnica de recomendación y esta produce un tipo de modelo, el cual luego será usado como la entrada para otra técnica de recomendación.

Burke en [17] indica que los recomendadores que combinan diferentes técnicas son los que mejor mantienen la promesa de resolver el problema de *inicio frio*. En su estudio también usa un dataset de restaurantes llamado *entree* para evaluar varios algoritmos de 4 diferentes técnicas y sus hibridaciones usando 6 de los 7 métodos de hibridación definidos en [32]. Observa que algunas hibridaciones funcionan mejor que otras para el conjunto de datos usado. Algunas inclusive tienen peor resultado que los algoritmos base de cada técnica. Indica que todo estudio está por naturaleza limitado por las particularidades de los datos y el dominio de recomendación.

Burke [17] aclara que pueden presentarse hibridaciones entre diferentes algoritmos de una misma técnica y entregar resultados que mejoren cada algoritmo individual.

ALGORITMOS DE FILTRO COLABORATIVO

A continuación, se presentan algunos algoritmos de la técnica de filtro colaborativo, que serán usados en el prototipo como conjunto de algoritmos para la evaluación y selección en cada conjunto de datos. Estos algoritmos son implementados en librerias como Surprise [33] de Python la cual será utilizada en la implementación de prototipo.

Algoritmos básicos

Normal Predictor: Este algoritmo predice una calificación aleatoria basándose en la distribución del conjunto de datos de entrenamiento, el cual se asume que es normal.

La predicción de la calificación $\hat{r}_{ui}$ es generada desde una distribución normal $\mathcal{N}(\hat{\mu}, \hat{\sigma}^2)$, donde $\hat{\mu}$ (1) y $\hat{\sigma}$ (2) son estimados desde el conjunto de entrenamiento usando la técnica *Maximum Likelihood Estimation*.

$$\hat{\mu} = \frac{1}{|R_{train}|} \sum_{r_{ui} \in R_{train}} r_{ui} \quad (1)$$

$$\hat{\sigma} = \sqrt{\sum_{r_{ui} \in R_{train}} \frac{(r_{ui} - \hat{\mu})^2}{|R_{train}|}} \quad (2)$$

Maximum Likelihood Estimation se define como un método para estimar los parámetros de la población desde los datos de la muestra, de tal forma que sea maximizada la probabilidad (*likelihood*) de obtener los datos observados.

Baseline only: Este algoritmo predice estimaciones de referencia para cada usuario y cada artículo.

Koren en [34] describe como en los datos se exhiben grandes efectos entre usuarios y entre artículos. Por ejemplo, las tendencias de algunos usuarios a dar calificaciones más altas que otros. De igual forma algunos artículos pueden tender a recibir calificaciones más altas que otros. Se acostumbra a ajustar los datos teniendo en cuenta estos efectos.

La estimación de referencia para una calificación desconocida r_{ui} (3), se denota como b_{ui} y tiene en cuenta los efectos del usuario y el artículo:

$$\hat{r}_{ui} = b_{ui} = \mu + b_u + b_i \quad (3)$$

μ es la calificación promedio general, los parámetros b_u y b_i indican la desviación observada para el usuario u y el artículo i respectivamente.

Si el usuario u es desconocido, b_u se asume como cero. Lo mismo aplica para el artículo con b_i.

Aproximación básica de k vecinos más próximos (KNN - K-Nearest Neighbors)

KNN Basic: Algoritmo de filtro colaborativo básico, la predicción de $\hat{r}_{ui}$ (4)(5) se presenta como:

$$\hat{r}_{ui} = \frac{\sum_{v \epsilon N_i^k(u)} sim(u,v).r_{vi}}{\sum_{v \epsilon N_i^k(u)} sim(u,v)} \text{ (Para basado en usuario).} \quad (4)$$

$$\hat{r}_{ui} = \frac{\sum_{j \epsilon N_u^k(i)} sim(i,j).r_{uj}}{\sum_{j \epsilon N_u^k(j)} sim(i,j)} \text{ (Para basado en artículo).} \quad (5)$$

KNN es un algoritmo de aprendizaje supervisado que clasifica cada elemento nuevo en un grupo, esta clasificación es determinada por el numero de vecinos más próximos que tenga de un grupo o de otro. La distancia entre elementos se calcula mediante medidas de similitud (sim), estas permiten encontrar que tan similares son dos elementos con relación a uno o varios aspectos. Por ejemplo, dos usuarios son muy similares si calificaron los mismos artículos con valores muy parecidos (altos o bajos). Algunas medidas de similitud son: similitud de coseno, similitud de Jaccard, similitud de Pearson, entre otras.

KNN with Means: Algoritmo de filtro colaborativo básico que tiene en consideración la calificación media de cada usuario. La predicción de $\hat{r}_{ui}$ (6)(7) se presenta como:

$$\hat{r}_{ui} = \mu_u + \frac{\sum_{v \epsilon N_i^k(u)} sim(u,v).(r_{vi} - \mu_v)}{\sum_{v \epsilon N_i^k(u)} sim(u,v)} \text{ (Para basado en usuario).} \quad (6)$$

$$\hat{r}_{ui} = \mu_i + \frac{\sum_{j \epsilon N_u^k(i)} sim(i,j).(r_{uj} - \mu_j)}{\sum_{j \epsilon N_u^k(j)} sim(i,j)} \text{ (Para basado en artículo).} \quad (7)$$

KNN with Z-Score: Algoritmo de filtro colaborativo básico que tiene en consideración la normalización Z-Score de cada usuario. La predicción de $\hat{r}_{ui}$ (8)(9) se presenta como:

$$\hat{r}_{ui} = \mu_u + \sigma_u \frac{\sum_{v \epsilon N_i^k(u)} sim(u,v).(r_{vi} - \mu_v)/\sigma_v}{\sum_{v \epsilon N_i^k(u)} sim(u,v)} \text{ (Para basado en usuario).} \quad (8)$$

$$\hat{r}_{ui} = \mu_i + \sigma_i \frac{\sum_{j \epsilon N_u^k(i)} sim(i,j).(r_{uj} - \mu_j)/\sigma_j}{\sum_{j \epsilon N_u^k(j)} sim(i,j)} \text{ (Para basado en artículo).} \quad (9)$$

KNN Baseline: Algoritmo de filtro colaborativo básico que tiene en consideración la calificación de referencia. La predicción de $\hat{r}_{ui}$ (10)(11) se presenta como:

$$\hat{r}_{ui} = b_{ui} + \frac{\sum_{v \epsilon N_i^k(u)} sim(u,v).(r_{vi} - b_{vi})}{\sum_{v \epsilon N_i^k(u)} sim(u,v)} \text{ (Para basado en usuario).} \quad (10)$$

$$\hat{r}_{ui} = b_{ui} + \frac{\sum_{j \epsilon N_u^k(i)} sim(i,j).(r_{uj} - b_{uj})}{\sum_{j \epsilon N_u^k(j)} sim(i,j)} \text{ (Para basado en artículo).} \quad (11)$$

Burke et al. [2] Habla sobre las ventajas de basado en artículo sobre basado en usuario. La versión basada en artículo para algoritmos de KNN ha mostrado mejor escalamiento y produce recomendaciones más exactas para grandes colecciones de artículos en el conjunto de datos.

Los algoritmos de KNN basados en memoria tienen dos limitaciones importantes computacionalmente [2]:

- Escalabilidad: La necesidad de comparar cada usuario con los demás del conjunto de datos hace que esas técnicas no sean prácticas para conjuntos de datos grandes.

- Sparsity (escasez de datos): La necesidad de comparar directamente las calificaciones de los artículos ocasiona que, en colecciones de datos con pocas calificaciones, los usuarios puedan tener muy pocos vecinos.

Algoritmos basados en factorización matricial

Se busca factorizar una matriz con el objetivo de encontrar dos o más matrices, que al multiplicarlas se obtenga de nuevo la matriz original.

La factorización matricial puede usarse para descubrir las características latentes que se encuentran dentro de un espacio latente de dimensionalidad f y que mejor describen las interacciones entre dos entidades (o más, aplicando factorización tensorial). En sistemas de recomendación, las entidades son artículos y usuarios. Estas características latentes determinan como un usuario califica un artículo. Por lo tanto, al descubrir estas características se puede predecir una calificación desconocida entre un usuario y un artículo.[35]

En factorización matricial cada artículo i es asociado con un vector $q_i \in \mathbb{R}^f$, el cual mide la cantidad que posee el artículo de cada factor, positivo o negativo. De igual forma, cada usuario u es asociado a un vector $p_u \in \mathbb{R}^f$ y este mide el interés que tiene el usuario en artículos con valores altos en cada factor, el interés puede ser positivo o negativo.

Matrix factorization SVD (Singular Value Decomposition):

El producto punto resultante entre ${q_i}^T p_u$ captura el interés en general que tiene un usuario sobre todas las características del artículo. En esta técnica la calificación final es creada agregando también los predictores de línea base mencionados anteriormente. Por lo tanto, una calificación se predice de la siguiente forma:

$$\hat{r}_{ui} = \mu + b_u + b_i + {q_i}^T p_u \tag{12}$$

Para aprender los parámetros del modelo ($b_u, b_i, p_u \; y \; q_i$) se minimiza el error cuadrado regularizado:

$$\min_{b_*, q_*, p_*} \sum_{(u,i)\in\mathcal{K}} (r_{ui} - \mu - b_i - b_u - q_i^T p_u)^2 + \lambda_4 (b_i^2 + b_u^2 + \|q_i\|^2 + \|p_u\|^2) . \tag{13}$$

La constante λ_4 controla el grado de regularización y normalmente se determina por validación cruzada, La minimización se puede realizar por SGD (*Stochastic Gradient Descent*) o por ALS (*Alternating Least Squares*) [36].

Es posible usar una versión del algoritmo que no hace uso de los predictores de línea base, lo cual es equivalente a la factorización matricial probabilística:

$$\hat{r}_{ui} = {q_i}^T p_u \tag{14}$$

Matrix factorization SVD++:

En esta técnica la precisión de la predicción es mejorada al considerar la retroalimentación implícita de los usuarios. Esto es útil para los usuarios que proveen más información implícita que explícita. Aun en casos donde es ausente la retroalimentación implícita, se puede capturar una señal significativa basándose en los artículos que el usuario califica, independiente del valor de la calificación. Para esto, se añade un segundo conjunto de factores del artículo, relacionando cada

artículo i a un vector de factores $y_i \in \mathbb{R}^f$. Estos nuevos factores de artículo se usan para caracterizar usuarios basándose en el conjunto de artículos que estos han calificado [2] [36]. El modelo es de la siguiente forma:

$$\hat{r}_{ui} = \mu + b_u + b_i + {q_i}^T \left(p_u + |R(u)|^{-\frac{1}{2}} \sum_{j \in R(u)} y_j\right) \quad (15)$$

$R(u)$ es el conjunto de artículos calificados por el usuario u.

El usuario u se modela como $p_u + |R(u)|^{-\frac{1}{2}} \sum_{j \in R(u)} y_j$. Se usa un vector libre de factores de usuario p_u tal y como se hace en SVD, el cual se aprende de las calificaciones explícitas del usuario. Este vector se complementa con la suma $|R(u)|^{-\frac{1}{2}} \sum_{j \in R(u)} y_j$, la cual representa la perspectiva de la retroalimentación implícita.

Los parámetros del modelo se encuentran minimizando la función regularizada del error cuadrático asociada por medio de Stochastic Gradient Descent [36].

Non-Negative Matrix factorization:

Este algoritmo es muy similar al SVD. Una calificación se predice de la siguiente forma:

$$\hat{r}_{ui} = {q_i}^T p_u \text{ (versión sin predictores de línea base)} \quad (16)$$

$$\hat{r}_{ui} = \mu + b_u + b_i + {q_i}^T p_u \text{ (versión con predictores de línea base)} \quad (17)$$

En este modelo, los factores de artículo y usuario se mantienen positivos mediante la manipulación diagonal de la tasa de aprendizaje, cancelando así los componentes negativos y por lo tanto, se reformula el proceso de aprendizaje [37].

El procedimiento de actualización es un Stochastic Gradient Descent (regularizado) con una selección específica del tamaño de cada paso de actualización que asegura la no negatividad de los factores, siempre que sus valores iniciales también sean positivos [33].

Algoritmos Slope one

Algoritmo de filtro colaborativo sencillo y preciso[33].

Se define la predicción de la calificación como:

$$\hat{r}_{ui} = \mu_u + \frac{1}{|R_i(u)|} \sum_{j \epsilon R_i(u)} dev(i,j) \quad (18)$$

Donde $R_i(u)$ es el conjunto de artículos relevantes, por ejemplo, el conjunto de artículos j calificados por u y que también tienen al menos un usuario en común con i.

$dev(i,j)$ en (19) se define como la diferencia promedio entre las calificaciones de i y las de j.

$$dev(i,j) = \frac{1}{|U_{ij}|} \sum_{u \epsilon U_{ij}} r_{ui} - r_{uj} \quad (19)$$

Co-clustering

Algoritmo de filtro colaborativo basado en co-clustering.

Básicamente los artículos y los usuarios son asignados a algunos clusters C_u , C_i y algunos co-clusters C_{ui}.

La predicción de la calificación está dada como [33]:

$$\hat{r}_{ui} = \overline{C_{ui}} + (\mu_u - \overline{C_u}) + (\mu_i - \overline{C_i}) \quad (20)$$

Donde $\overline{C_{ui}}$ es la calificación promedio del co-cluster C_{ui}, $\overline{C_u}$ es la calificación promedio del cluster C_u, y $\overline{C_i}$ es la calificación promedio del cluster C_i

Si el usuario es desconocido, la predicción es $\hat{r}_{ui} = \mu_i$. Si el artículo es desconocido, la predicción es $\hat{r}_{ui} = \mu_u$. Si el usuario y el artículo son desconocidos, la predicción es $\hat{r}_{ui} = \mu$.

MÉTRICAS DE EVALUACIÓN PARA UN SISTEMA DE RECOMENDACIÓN

Al momento de diseñar un sistema de recomendación, se encuentran disponibles una gran variedad de algoritmos y se debe determinar cuál cumple mejor con los objetivos del sistema a desarrollar. Estas decisiones se toman comúnmente basándose en experimentos y en la comparación del rendimiento entre los algoritmos candidatos.

Shani et al. [21] basándose en protocolos de evaluación usados en áreas como aprendizaje de máquinas y recuperación de información, describe tres niveles de experimentación que pueden ser usados para comparar recomendadores, estos son:

- Experimentos offline: Son los más fáciles de realizar ya que no requieren interacción con usuarios reales y esto permite comparar un amplio rango de algoritmos candidatos a bajo costo. Se usan datos recolectados con anterioridad que describen la selección o calificación que han dado un conjunto de usuarios a diferentes artículos. Usando estos datos se trata de simular los comportamientos de los usuarios que interactúan con el sistema de recomendación. Con este tipo de experimento solo se pueden responder algunas preguntas, principalmente relacionadas con el poder de predicción de un algoritmo, pero no se puede medir directamente la influencia del recomendador sobre los comportamientos del usuario.

- Estudios de usuario: Se pide a un pequeño grupo de sujetos que usen el sistema en un entorno controlado y luego reporten su experiencia. En estos experimentos se puede recoger información cualitativa y cuantitativa del sistema, pero se deben considerar varios sesgos (biasses) en el diseño del experimento.
- Experimentos online: Es quizás uno de los experimentos más confiables ya que se usa el sistema con usuarios reales que realizan tareas reales. Estos usuarios típicamente no están conscientes del experimento. Esto hace que el diseño del experimento esté más cerca de la realidad. Este experimento permite medir el cambio en los comportamientos del usuario cuando estos interactúan con diferentes sistemas de recomendación, por ejemplo, si los usuarios de un sistema siguen las recomendaciones más a menudo o si la utilidad obtenida por los usuarios de un sistema excede la utilidad obtenida por los usuarios de otro sistema, entonces se concluye que un sistema es superior al otro.

Inicialmente la mayoría de recomendadores se evaluaban y puntuaban con respecto a su poder de predicción (su habilidad de predecir con precisión las elecciones del usuario). Sin embargo, en la actualidad es ampliamente aceptado que las predicciones precisas son fundamentales pero insuficientes para desplegar un buen sistema de recomendación. Los usuarios del sistema, además de recibir recomendaciones, pueden tener interés en descubrir nuevos y diversos artículos [27], en preservar su privacidad, en la rápida respuesta del sistema, entre otros [21].

Propiedades de un sistema de recomendación

Shani et al. [21] describe las propiedades que son comúnmente consideradas al momento de decidir cuál enfoque de recomendación utilizar. Según las necesidades del sistema, se debe decidir cuáles de estas propiedades se deben medir y en algunos casos debe considerarse un compromiso o intercambio entre las propiedades del sistema, por ejemplo, renunciar a algo de exactitud para mejorar la

diversidad de las recomendaciones. Algunas de estas propiedades deben evaluarse por medio de un estudio de usuario o un experimento online para ver cómo los cambios en esa propiedad afectan realmente la experiencia del usuario.

1. Preferencia del usuario: Basándose en el problema de seleccionar un algoritmo entre una lista de candidatos, una opción obvia es realizar un estudio de usuarios en el que se le pida a los participantes que seleccionen uno de los sistemas según su experiencia. Esta opción asume que todos los usuarios son iguales y esto no siempre es cierto, por ejemplo, un usuario que usa constantemente el sistema deberá tener una opinión más valiosa que la de un usuario que solo uso el sistema una vez. Por lo tanto es necesario asignar pesos de importancia a los votos de los usuarios y esto no es una tarea fácil en un estudio de usuario.

   Otro caso importante al medir la preferencia del usuario es cuando los usuarios que prefieren el sistema A, solo lo prefieren un poco, mientras que los usuarios que prefieren B, tienen una opinión muy baja de A. en este caso, aun cuando más usuarios prefieran A sería una mejor opción seleccionar B. Para medir lo anterior es necesario que las respuestas a las preguntas de preferencia en el estudio de usuario no sean binarias.

   Más allá de conocer la preferencia total de un usuario para un sistema sobre los demás sistemas, es importante separar las propiedades del sistema y conocer la preferencia del usuario hacia esas propiedades específicas.
2. Precisión de predicción: Es la propiedad más discutida en la literatura de los sistemas de recomendación [2]. Medir la precisión de la predicción es independiente de la interface del usuario, por lo tanto, puede ser medida en un experimento *offline*.

   Se presentan tres clases para medir la precisión de la predicción:

   a. Medir la precisión en la predicción de calificaciones: Medir que tan bien el sistema predice la calificación que un usuario particular le dará a un artículo.

i. Root Mean Squared Error (RMSE): Es la métrica más popular en la evaluación de la precisión prediciendo ratings. El sistema predice los ratings $\hat{r}_{ui}$ para un conjunto de prueba $\mathcal{T}$ que contiene pares de usuario-artículo (u, i) para los cuales se conoce la calificación verdadera r_{ui}. El RMSE entre la calificación predecida y la calificación verdadera está dado por:

$$RMSE = \sqrt{\frac{1}{|\mathcal{T}|} \sum_{(u,i)\in\mathcal{T}} (\hat{r}_{ui} - r_{ui})^2} \quad (21)$$

ii. Mean Absolute Error (MAE): Es una alternativa popular y está dada por:

$$MAE = \sqrt{\frac{1}{|\mathcal{T}|} \sum_{(u,i)\in\mathcal{T}} |\hat{r}_{ui} - r_{ui}|} \quad (22)$$

La diferencia entre RMSE y MAE es que RMSE penaliza los errores grandes, por lo tanto, para un conjunto de prueba con 4 artículos ocultos, RMSE prefiere un sistema con un error de 2 en tres artículos y un error de 0 en el cuarto artículo. Por el contrario, MAE prefiere un sistema que tenga un error de 3 en un artículo y un error de 0 en los otros tres artículos.

iii. RMSE Normalizado (NRMSE) y MAE Normalizado (NMAE): Son versiones de RMSE y MAE que han sido normalizadas por el rango de calificaciones, por ejemplo, $(r_{max} - r_{min})$. Simplemente son versiones escaladas de RMSE y MAE, por lo tanto, el resultado de ranking para los algoritmos será el mismo que el de las medidas sin normalizar.

iv. RMSE promedio y MAE promedio: Esta medida permite ajustar conjuntos de prueba no balanceados, por ejemplo, los valores de

RMSE y MAE obtenidos están altamente influenciados por el error de unos pocos artículos que son muy frecuentes.

b. Medir la predicción de uso: En muchas aplicaciones de sistemas de recomendación, el sistema no busca predecir la preferencia de los usuarios hacia los artículos, por ejemplo, la calificación de un usuario para un artículo. Estos sistemas buscan recomendar a los usuarios, artículos que estos puedan usar.

En una evaluación offline de la predicción de uso, se tiene un conjunto de datos que contienen los artículos que cada usuario ha usado. Se selecciona un usuario de prueba y se ocultan algunas de sus selecciones, luego se le pide al sistema de recomendación que prediga un conjunto de artículos que el usuario usará. Ante lo anterior se pueden presentar cuatro posibles salidas para los artículos recomendados y ocultos, *Tabla 1*.

	Recomendado	**No recomendado**
Usado	True-Positive (tp)	False-Negative (fn)
No usado	False-Positive (fp)	True-Negative (tn)

Tabla 1: Matriz de confusión. Tomada de [21]

En el caso de evaluación *offline* se debe asumir que los artículos que no han sido usados por el usuario en el conjunto de datos, no serán usados aun si son recomendados (no son de interés para el usuario). Esta suposición puede ser falsa ya que los artículos recomendados que no han sido usados por el usuario, puede que, si sean de su interés, pero el usuario no conocía su existencia. En este caso el número de falsos positivos estaría sobreestimado.

Contando el número de ejemplos que caen en cada celda de la tabla 1, se pueden calcular las siguientes cantidades:

$$Precision = \frac{\#tp}{\#tp + \#fp}$$

$$Recall\ (True\ Positive\ Rate) = \frac{\#tp}{\#tp + \#fn}$$

$$False\ Positive\ Rate\ (1 - Specificity) = \frac{\#fp}{\#fp + \#tn}$$

Se puede esperar una compensación entre las cantidades anteriores, por ejemplo, si se entrega una lista de recomendaciones más grande, se mejoraría el *Recall*, pero esto también reduciría la precisión de las recomendaciones.

Precisión y *recall* son las métricas más populares en el campo de la recuperación de información [38].

Es posible calcular curvas que comparen precisión y *recall*, conocidas como curvas precisión-*recall*. También se pueden calcular curvas que comparen True Positive Rate y False Positive Rate, estas son conocidas como Receiver Operating Characteristic o curvas ROC. Las curvas precisión-recall enfatizan en la proporción de artículos recomendados que son preferidos y las curvas ROC enfatizan en la proporción de artículos que no son preferidos y que son recomendados.

Para comparar dos algoritmos, se puede calcular un par de estas curvas, una para cada algoritmo. Si una de estas curvas domina completamente la otra, la decisión acerca del algoritmo superior es sencilla. Sin embargo, cuando las curvas se intersectan, la decisión es menos obvia y dependerá de la aplicación en cuestión, la selección de la región de la curva en la que se basará la decisión.

Medidas como F-measure y Area Under the ROC Curve (AUC) son útiles para comparar algoritmos independientemente de la aplicación, pero es preferible tomar decisiones basadas en medidas que reflejen las necesidades específicas de la aplicación [21].

Las métricas de exactitud en la clasificación permite evaluar qué tan efectivamente las predicciones ayudan al usuario a distinguir buenos ítems de malos ítems. Seleccionar métricas de exactitud en la clasificación es útil cuando no hay interés en el valor exacto de predicción, pero si en averiguar si al usuario activo le gustarán o no los elementos presentados en las recomendaciones [38].

c. Medidas de Ranking (clasificación): En muchas aplicaciones se presenta al usuario una lista de recomendaciones, en esas aplicaciones no hay interés en predecir una calificación o seleccionar un conjunto de artículos a recomendar, pero si en ordenar los artículos en la lista de acuerdo con las preferencias del usuario. A esta tarea se le conoce como Ranking de artículos. Existen dos enfoques para medir la exactitud de este Ranking
    i. Usar un Ranking de referencia: Se busca evaluar un algoritmo de ranking con respecto a un ranking de referencia (un orden correcto), para esto es necesario obtener tal referencia. Por ejemplo, la lista de artículos ordenada según el grado de interés para un usuario.
    ii. Ranking basado en la utilidad: Se asume que la utilidad de una lista de recomendaciones está dada por la suma de las utilidades de cada recomendación individual que se encuentra en la lista.

   La utilidad de cada recomendación es la utilidad del artículo recomendado descontada por un factor que depende de la posición en la lista.

3. Cobertura: El término cobertura se refiere a las siguientes propiedades del sistema:
    a. Cobertura del espacio de artículos: Se refiere a la proporción de artículos que el sistema de recomendación puede recomendar, esto se refiere a la cobertura del catálogo de artículos disponibles, la medida más simple para esto es el porcentaje de todos los artículos que pueden ser recomendados al menos una vez.

    Una medida más útil es el porcentaje de todos los artículos que son recomendados durante un experimento ya sea offline, online o estudio de usuario. En algunos casos es deseable asignar pesos a cada artículo, por ejemplo de acuerdo a su popularidad o utilidad, para luego decidir si esta bien que el sistema no recomiende ciertos artículos que son poco usados y tener menos tolerancia con los artículos muy populares que son ignorados.

    b. Cobertura del espacio de usuarios: La cobertura en este caso puede ser la proporción de usuarios o interacciones de usuario para los cuales el sistema puede recomendar artículos. En algunas aplicaciones el sistema no provee recomendaciones para ciertos usuarios por motivos como poca confianza en la exactitud de las predicciones para ese usuario. En estos casos son preferibles los sistemas que puedan proveer recomendaciones para un amplio rango de usuarios. Estos sistemas deberían ser evaluados en el compromiso entre cobertura y exactitud de las recomendaciones.
    c. Cold Start (inicio frio): El desempeño del sistema para nuevos artículos o nuevos usuarios. Es considerado un sub-problema de cobertura ya que este mide la cobertura del sistema para un conjunto específico de usuarios o artículos.

4. Confianza del sistema: Se refiere a la confianza que tiene el sistema en sus recomendaciones o predicciones [21]. Por ejemplo, filtro colaborativo mejora la exactitud de sus recomendaciones a medida que crece la cantidad de

datos sobre los artículos. De forma similar la confianza en la predicción aumenta cuando hay más datos.

En muchos casos los usuarios se pueden beneficiar al conocer las puntuaciones de confianza, por ejemplo, si el sistema reporta un bajo nivel de confiabilidad en un artículo recomendado, el usuario puede investigar más sobre ese artículo antes de usarlo, mientras que, si el sistema reporta un alto nivel de confianza en una recomendación, el usuario puede usar el artículo recomendado inmediatamente.

5. Confianza del usuario: Se refiere a la confianza que tiene el usuario en las recomendaciones que le entrega el sistema. Por ejemplo, es beneficioso para el sistema, recomendar artículos que el usuario ya conoce y le han gustado. De esta forma, el usuario no obtiene ningún valor en esas recomendaciones, pero observa que el sistema ofrece recomendaciones razonables para sus preferencias, lo cual aumentará la confianza del usuario en el sistema cuando este le recomiende artículos desconocidos.

   Otra forma común de mejorar la confianza del usuario en el sistema es explicando las recomendaciones que el sistema provee. El método para evaluar la confianza del usuario en el sistema es preguntando a los usuarios si las recomendaciones entregadas por el sistema son razonables en un estudio de usuario [21].

6. Novedad: Se refiere a recomendaciones para artículos que el usuario no conocía. En aplicaciones que requieren novedad en las recomendaciones, la forma más sencilla es sacar los artículos que el usuario ya ha usado o calificado. En muchos casos los usuarios no reportan todos los artículos que han usado en el pasado, por lo tanto, el método anterior es insuficiente.

   Se puede medir la novedad en un estudio de usuario, preguntando a los usuarios si ya están familiarizados con un artículo recomendado. De igual forma se puede ganar un entendimiento sobre la novedad del sistema

mediante un experimento offline en el que se partirá el conjunto de datos en el tiempo, por ejemplo, ocultar todas las calificaciones del usuario que ocurrieron después de un punto específico en el tiempo y adicionalmente ocultar algunas calificaciones que ocurrieron antes de ese momento, simulando los artículos con los que el usuario está familiarizado pero que no ha reportado al sistema. Al momento de recomendar, el sistema es premiado por cada artículo recomendado que haya sido calificado después del punto en el tiempo definido, y será castigado por cada artículo recomendado que el usuario haya usado antes del punto definido.

7. Serendipia: Es la medida de que tan sorprendentes son las recomendaciones exitosas. Por ejemplo, si un usuario ha calificado positivamente varias películas en las que se encuentra cierto actor, recomendar una nueva película de ese actor sería novedoso, porque el usuario no conocía esa película, pero no es completamente sorprendente. Por otro lado, recomendaciones aleatorias serían muy sorprendentes, pero es necesario balancear serendipia con exactitud. Se puede entender serendipia como la cantidad de información relevante que es nueva para el usuario en una recomendación. Por ejemplo, si al usuario seguir una recomendación de una película, el usuario aprende sobre un nuevo actor que le gusta. Existen técnicas de recuperación de información que pueden aplicarse a sistemas de recomendación cuando hay disponibles metadatos sobre los artículos, como información de contenido [21].

8. Diversidad: Generalmente definida como lo opuesto a similitud. En algunos casos sugerir al usuario una lista de artículos similares no podría ser los suficientemente útil para este, ya que le tomaría tiempo explorar entre los artículos propuestos.
   Los métodos más explorados para medir la diversidad usan la similitud entre artículos, generalmente basándose en el contenido de estos.

La diversidad puede venir a expensas de otras propiedades como la exactitud, se pueden calcular curvas para evaluar la reducción en la exactitud vs. El incremento en diversidad [21].

9. Utilidad: Algunas aplicaciones de los sistemas de recomendación, como los sitios web de e-commerce, buscan generar ganancias por medio de las recomendaciones generadas. Para estos sistemas, medir la utilidad, o la utilidad esperada de las recomendaciones puede ser más significativo que medir la exactitud de las recomendaciones. Estos sistemas deben definir funciones de utilidad, ya sea basándose en otras propiedades como diversidad o serendipia, o en el valor que ganan tanto los usuarios como el sistema de las recomendaciones generadas [21].

10. Riesgo: En algunos casos una recomendación puede estar asociada con un riesgo potencial. Por ejemplo, cuando se recomiendan acciones para comprar, algunos usuarios tienen aversión al riesgo y prefieren acciones con poco crecimiento esperado, pero que a su vez tienen poco riesgo de colapsar. Por otro lado, algunos usuarios prefieren tomar riesgos y buscar acciones con mayor probabilidad de ganancia.
    La forma estándar de evaluar sistemas sensibles al riesgo es considerando no solo la utilidad esperada, sino también la varianza de la utilidad [21].

11. Robustez: Se refiere a la estabilidad de las recomendaciones en la presencia de información falsa, típicamente insertada a propósito con el objetivo de influenciar las recomendaciones para que estas guíen a los usuarios hacia ciertos artículos que generen mayores ganancias a los interesados. Estos intentos de influenciar las recomendaciones son llamados ataques.
    Otro tipo de robustez se refiere a la estabilidad del sistema bajo condiciones extremas, como un gran número de peticiones [21].

12. Privacidad: En un sistema de recomendación, un usuario revela sus preferencias hacia los artículos del sistema con la esperanza de obtener recomendaciones útiles. Sin embargo, para la mayoría de los usuarios es importante que sus preferencias se mantengan privadas, esto es, que ningún tercero pueda usar el sistema de recomendación para aprender sobre el usuario. Por esta razón el análisis de la privacidad tiende a enfocarse en el peor escenario, ilustrando casos teóricos en los que la información privada de los usuarios podría ser revelada.

    No es realista tratar de conseguir la privacidad completa en un sistema de recomendación, pero es muy importante minimizar lo máximo posible las brechas en la privacidad [21].

13. Adaptabilidad: Los sistemas de recomendación reales pueden operar en un entorno en donde la colección de artículos cambia rápidamente, o donde las tendencias en el interés sobre los artículos pueden cambiar.

    En el caso donde los artículos cambian rápidamente, un ejemplo sería un recomendador de noticias, en donde las historias nuevas son interesantes solo por un corto periodo de tiempo o donde las noticias viejas pueden volverse interesantes repentinamente cuando ocurre un evento relacionado. Este tipo de adaptación puede ser evaluada offline, analizando la cantidad de información requerida antes de que un artículo sea recomendado: Lo anterior puede requerir que el proceso de recomendación sea modelado de manera secuencial.

    En el otro caso, el sistema debe adaptarse a las preferencias personales del usuario o a cambios en el perfil del usuario. Por ejemplo, cuando un usuario ingresa una calificación para un artículo, se espera que cambie la lista de recomendaciones que se le presenta a ese usuario. Si la lista se mantiene igual, el usuario asumirá que no vale el esfuerzo de calificar artículos.

Es posible evaluar mediante un experimento offline, los cambios en las listas de recomendaciones después de agregar más información al perfil del usuario, por ejemplo, nuevas calificaciones [21].

14. Escalabilidad: Una de las metas en el diseño de sistemas de recomendación es que estos escalen a conjuntos de datos reales, los cuales pueden crecer en el tiempo y ocasionar que el algoritmo sea más lento o que se requieran recursos computacionales adicionales.

    La escalabilidad se mide típicamente mediante la experimentación con conjuntos de datos en crecimiento, mostrando cómo se comportan la velocidad y el consumo de recursos a medida que la tarea aumenta.

    En muchos casos se espera que los sistemas de recomendación provean de forma rápida las recomendaciones online. Una medida para esto es el rendimiento o throughput del sistema, por ejemplo, el número de recomendaciones que el sistema provee por segundo. También se puede medir la latencia o tiempo de respuesta, que indica el tiempo requerido para hacer una recomendación online [21].

ETL (EXTRACT – TRANSFORM – LOAD)

ETL hace referencia a las herramientas o piezas de software que son responsables de la extracción de los datos desde diferentes fuentes, su limpieza, personalización e inserción en un almacén de datos (data warehouse) [39] u otros repositorios de datos.

Un sistema de recomendación requiere de datos representados como interacciones para alimentar los algoritmos que permiten predecir la preferencia o calificación de un usuario hacia un artículo. Estos datos pueden ser tomados desde diferentes fuentes y encontrarse en diferentes formatos. Es por esto necesario la aplicación de

un proceso ETL que permita la obtención, preparación y carga de los datos, para así ser consumidos por el resto del sistema.

Vassiliadis et al. [39] presentan un framework general para el proceso ETL en la *Figura 2*.

En la capa inferior se representan los almacenamientos de datos que estarán involucrados en el proceso completo. En el lado izquierdo de la capa inferior se encuentran las fuentes originales de datos, estos datos son extraídos por medio de rutinas de extracción que pueden obtener los datos completos en una iteración o pueden obtener diferenciales de los datos en varias iteraciones (capa superior - izquierda). Los datos extraídos son propagados a un DSA (Data Stage Area) o área de pruebas, la cual es un área de almacenamiento intermedia entre la fuente de los datos y su destino, es allí donde son transformados. Por último los datos transformados son cargados al almacén de datos por medio de actividades de carga, como se muestra en la parte superior derecha de la figura 2. En la parte inferior derecha se representan los almacenamientos objetivo de los datos.

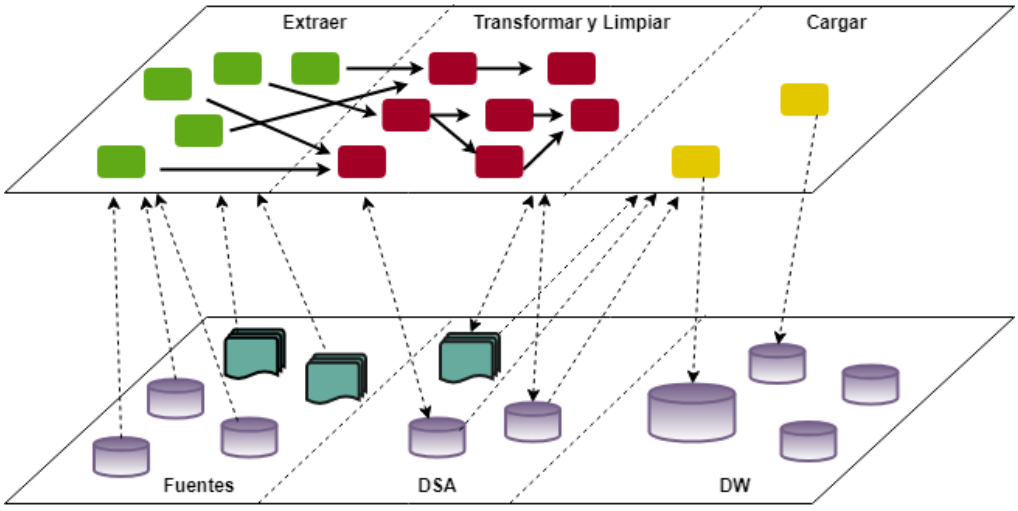

Figura 2: Framework general para el proceso ETL. Tomado de [39]

MÉTODO K-FOLD PARA VALIDACIÓN CRUZADA

La validación cruzada es un proceso de remuestreo usado para evaluar modelos de aprendizaje de máquina con una muestra de datos limitada. Este proceso permite estimar la habilidad del modelo con datos que nunca ha visto.

El método k-fold es una variación del método holdout. Holdout separa un conjunto de datos en dos partes (estas pueden tener diferentes proporciones): una de entrenamiento y otra de prueba. En sistemas de recomendación, este método toma aleatoriamente algunas calificaciones de un porcentaje de usuarios o de todos los usuarios y los usa como el conjunto de evaluación, dejando todas las demás calificaciones como el conjunto de entrenamiento.

K-fold usa un único parámetro llamado k, este se refiere al número de particiones que se harán sobre los datos de muestra. El proceso general de este método es el siguiente:

1. Revolver los datos aleatoriamente.
2. Separar los datos en k grupos.
3. Cada grupo se selecciona una única vez y se usa como conjunto de pruebas. (un grupo solo se usa una vez para pruebas y k-1 veces para entrenamiento).
4. Los demás k – 1 grupos se usan como conjunto de entrenamiento del modelo.
5. Se toman en cuenta las k evaluaciones del modelo para resumir la habilidad de este en la métrica evaluada.

Este método es popular ya que es simple de entender y los resultados son generalmente menos sesgados y menos optimistas que otros métodos, como una simple partición del conjunto de datos en entrenamiento y pruebas.

ARQUITECTURA DE REFERENCIA

Una arquitectura de referencia es una colección de mejores prácticas para un dominio determinado. Actúa como una guía para construir aplicaciones. También es definida como un vocabulario común que tiene el objetivo de estandarizar implementaciones independientes.[40]

La mayoría de las investigaciones realizadas sobre sistemas de recomendación se han enfocado en los algoritmos y la evaluación de diferentes métricas sobre las recomendaciones obtenidas y la experiencia del usuario que las recibe. Existen pocas investigaciones que propongan una arquitectura de referencia para sistemas de recomendación.

Sheth et al. [40] presentan una arquitectura de referencia para sistemas de recomendación sociales que fue diseñada para ser agnóstica a la aplicación y al dominio. Esta arquitectura fue aplicada en tres proyectos no relacionados que buscan proveer sugerencias a usuarios de cómo usar mejor determinadas herramientas de software. La arquitectura de referencia está formada por tres capas: Usuarios, seguridad – privacidad y base de datos. Y tres componentes: observador (Watcher) - observa las actividades del usuario -, aprendiz (Learner) – Infiere patrones y recomendaciones - y tutor (Advisor) – Provee las recomendaciones a los usuarios. Cada módulo contiene una lista de tareas que debe realizar y una lista de posibles plugins para funcionalidades adicionales. Para más detalles referir a [2, Fig 1]

Amatriain et al. [41] Describen como direccionaron en Netflix los retos asociados a crear una arquitectura de software que maneje grandes volúmenes de datos, sea sensible a las interacciones del usuario y permita experimentar fácilmente con nuevos enfoques de recomendaciones. La arquitectura propuesta se divide en tres tipos de ejecución: Online, Offline y Nearline. La ejecución online responde peticiones en tiempo real y tiene más limitaciones computacionales. La ejecución

offline tiene menos limitaciones computacionales ya que se ejecuta en batch, sin embargo, los datos más recientes no son tenidos en cuenta. La ejecución Nearline es un compromiso intermedio entre la ejecución online y offline, se pueden realizar cálculos Online que no requieren ser entregados en tiempo real.

Amatriain et al. [41] Señala dos problemas importantes a resolver en el desarrollo de la arquitectura propuesta: como combinar las señales y modelos necesarios en los diferentes tipos de ejecución y cómo combinar los resultados de recomendaciones intermedias y así tengan sentido para el usuario.

Lops et al. [8] propone una arquitectura de alto nivel para los recomendadores basados en contenido. Está compuesta por tres componentes principales que manejan una parte del proceso de recomendación. El primer componente es el analizador de contenido, el cual hace un tipo de pre-procesamiento para extraer información relevante estructurada en una forma adecuada que sea usable por los próximos componentes del proceso, usualmente se refiere a tecnicas de extracción de caracteristicas sobre información que no tiene estructura, por ejemplo, texto.

El segundo componente es el aprendizaje de perfil, el cual se encarga de recolectar los datos que representan las preferencias del usuario e intenta generalizar estos datos para construir el perfil del usuario. Por lo general, la estrategia de generalización se realiza a través de técnicas de aprendizaje automático. Por último el componente de filtrado, hace uso del perfil del usuario para sugerir artículos relevantes que coincidan con la representación del perfil.

Mohamed et al. en [42] propone una arquitectura técnica (Figura 3) en la que explica las interacciones del sistema de recomendación propuesto para proveedores de servicios o productos. En este sistema se busca solo recomendar los proveedores al usuario según la similitud entre sus perfiles y se asume que cada proveedor tiene su propio sistema de recomendación para sugerir sus productos o servicios.

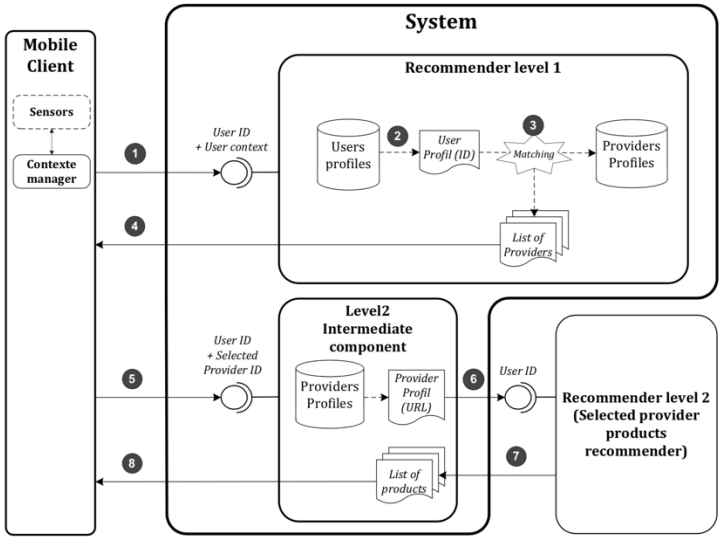

Figura 3: Arquitectura técnica para recomendar proveedores. Tomada de [42].

Esta arquitectura es muy específica para el problema que quiere tratar el autor y se enfoca en el dominio de los clientes móviles y los proveedores de servicios.

Narducci et al. En [43] proponen una arquitectura independiente del dominio (Figura 4) que describe el uso de análisis de emociones desde el texto para usar los resultados en la actualización del perfil de usuario y con este alimentar el sistema de recomendación

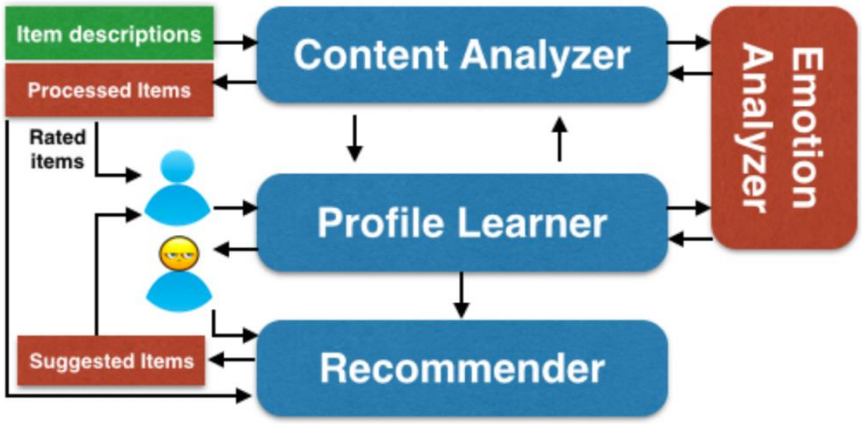

Figura 4: Arquitectura para sistema de recomendación con análisis de emociones. Tomado de [43].

El análisis de emociones está por fuera del alcance de este trabajo. Sin embargo, se considera una fuente de información interesante y se propone su adaptación a esta propuesta en trabajos futuros.

SELECCIÓN DE ALGORITMOS

Diferentes investigaciones en aprendizaje de máquina para sistemas de recomendación buscan identificar los algoritmos óptimos, Aquí, la optimalidad se define en términos de la máxima precisión alcanzable (posiblemente ponderada por la eficiencia computacional) con respecto a las metas de recomendación, configuración y restricciones específicas del dominio [16].

La decisión de cuál es el mejor algoritmo de recomendación para un problema específico es un tema importante y poco estudiado. Una tendencia común para resolver este problema es la evaluación experimental de varios algoritmos de recomendación en un conjunto de datos. Estos estudios requieren una gran cantidad de recursos computacionales y tiempo de procesamiento.

Brazdi et al. [44] presenta un método de meta-aprendizaje que permite seleccionar un algoritmo de aprendizaje. Este método usa un algoritmo KNN para identificar los conjuntos de datos más similares al conjunto de datos que se quiere analizar, esto mediante un conjunto pequeño de características de los datos. Estas características a su vez representan propiedades que afectan el rendimiento de los algoritmos. El rendimiento es juzgado mediante varias métricas de evaluación, el tiempo de ejecución es una de ellas.

Adomavicius et al. [45] indica que las calificaciones de un conjunto de datos pueden ser altamente escasas y sesgadas y esto puede tener un impacto significativo en el rendimiento de un sistema de recomendación. De igual forma, un algoritmo puede demostrar mejor rendimiento en un conjunto de datos, pero no en otros. Adomavicius et al. en [45] investiga cuáles características de los datos, conformados por calificaciones, están asociadas con la variación del rendimiento en las recomendaciones que entregan diferentes algoritmos de recomendación. Características como la densidad de los datos demostraron tener un efecto positivo en la exactitud de todos los algoritmos evaluados, mientras que la desviación estándar de las calificaciones tuvo un impacto negativo. Por otro lado, características como la forma del espacio de las calificaciones y la frecuencia en la distribución de las calificaciones, demostraron tener efectos positivos en unos algoritmos y efectos negativos en otros. No está en el alcance de este trabajo, estudiar las relaciones en las características de los datos con el rendimiento de los algoritmos, pero esta información permite identificar la necesidad de selección de un algoritmo que mejores resultados entregue con respecto a cada conjunto de datos específico. De igual forma, es relevante para el posterior análisis de los resultados obtenidos en la evaluación del prototipo.

Griffith et al. [46] extrae información de las calificaciones que describa al usuario en un conjuntos de datos de filtro colaborativo y usa un enfoque de aprendizaje de máquina supervisado para identificar si hay una relación entre esta información y la exactitud esperada en las recomendaciones del sistema. Las características

encontradas no son las mismas para todos los conjuntos de datos usados, pero se encuentran tendencias en las características seleccionadas y algunas coinciden en varios conjuntos de datos. Por ejemplo, el número de calificaciones, la desviación estándar de las calificaciones, el número de vecinos del usuario y la cantidad de artículos populares que califica. Esta aproximación busca relaciones en la información del usuario con el rendimiento de los algoritmos, se encuentran resultados muy similares a los de Adomavicius et al. en [45].

El alcance de este trabajo no busca encontrar un modelo de aprendizaje de máquina que permita seleccionar el algoritmo óptimo ya que se requieren muchos conjuntos de datos de ejemplo para entrenar tal modelo. Este tipo de enfoques se recomienda para trabajos futuros sobre el sistema propuesto.

CAPÍTULO 3: DISEÑO DE LA SOLUCIÓN

Esta sección describe el diseño del sistema propuesto y los detalles de implementación del prototipo. El sistema de recomendación presenta diferentes procesos necesarios para obtener los resultados o recomendaciones en un conjunto de datos específico: obtención del conjunto de datos, separación de los datos para entrenamiento y prueba de los algoritmos, entrenamiento y evaluación de los algoritmos, son las actividades comunes en un sistema de recomendación. La hibridación de algoritmos permite combinar las fortalezas de dos algoritmos y probar si los resultados mejoran los algoritmos base.

El sistema de recomendación interactúa con una entidad llamada *proveedor*, este hace referencia al cliente o negocio que requiere generar recomendaciones para sus usuarios. El *proveedor* debe recolectar y entregar al sistema la información requerida por los algoritmos a emplear. De igual forma, el proveedor se encarga de pedir y recibir las recomendaciones generadas por el sistema, para luego entregarlas a cada usuario específico. La interacción entre el sistema y el proveedor se presenta por medio de una capa de servicios que permiten el consumo de las recomendaciones y la definición de parámetros de configuración.

Se busca que el sistema propuesto pueda ser usado por diferentes proveedores, esto requiere la capacidad de usar fuentes de datos heterogéneas con diferentes formatos y estructura. Para esto se propone un método de estandarización de los datos que permite transformarlos a un formato y estructura común, el cuál puede ser consumido por los algoritmos presentes en el sistema.

Otro reto presente al momento de buscar un sistema que pueda ser usado por diferentes proveedores, se encuentra al momento de seleccionar cuál algoritmo es más óptimo para el conjunto de datos específico del proveedor. Algunas investigaciones en la literatura demuestran que las características del conjunto de

datos determinan el rendimiento de los diferentes algoritmos de recomendación (Sección 2.8). Se propone un modelo que permite priorizar los algoritmos de recomendación presentes en el sistema, mediante una combinación lineal que tiene en cuenta el resultado en 5 métricas de evaluación y el peso asociado a cada una de éstas.

La arquitectura del sistema propuesto se presenta en la *Figura 5*. En esta se puede ver la interacción de un *proveedor* con el sistema de recomendación mediante una *capa de servicio*. El *proveedor* envía los archivos que contienen los datos con los que se alimentarán los algoritmos de recomendación. Estos archivos son almacenados en un *área de archivos* definida para cada proveedor específico en el sistema.

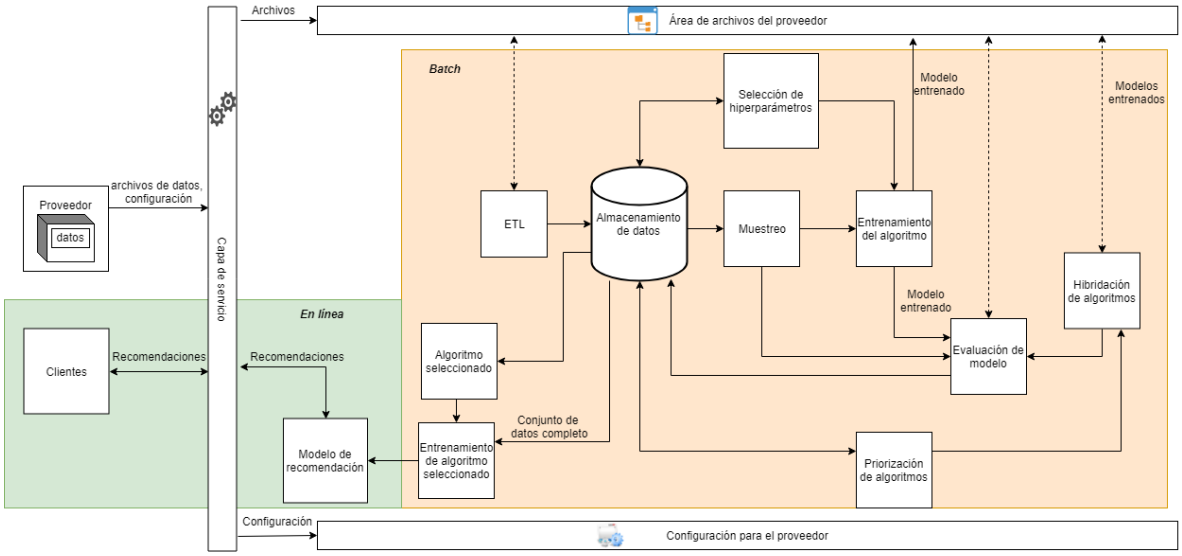

Figura 5: Arquitectura del sistema

El *proveedor* también puede, por medio de la *capa de servicio*, modificar los valores por defecto en la operación del sistema y es requerido que defina las reglas de mapeo para los datos que ingresa al sistema. Esta información se almacena en un archivo de configuración sobre el *área de archivos del proveedor*.

Los archivos del proveedor pasan por un proceso ETL en el que se estandarizan los datos (se describe en la sección 3.1) y la información queda almacenada de forma definitiva en una base de datos para el proveedor en el sistema, conformando así un conjunto de datos listo para ser usado por los algoritmos de recomendación. Los datos almacenados se van actualizando a medida que el proveedor entregue al sistema archivos con nuevos registros.

Algunos algoritmos de recomendación emplean hiperparámetros que determinan la configuración del modelo empleado por cada algoritmo y por lo tanto definen su capacidad y características de aprendizaje. El sistema propuesto contiene un módulo de selección de hiperparámetros, el cual se encarga de encontrar la configuración óptima para cada algoritmo mediante la minimización de una función de error como RMSE o MAE. Encontrar los hiperparámetros es un proceso costoso computacionalmente. Por lo tanto, existe una configuración base recomendada para cada algoritmo y es la que se usa por defecto. El uso de la selección de hiperparámetros es opcional y se define en la configuración del proveedor, esta se ejecuta de forma automática y hace uso del conjunto de datos del proveedor. Los hiperparámetros seleccionados o por defecto, son almacenados en la base de datos para el proveedor en el sistema. Esta configuración es usada por los algoritmos en la etapa de entrenamiento.

El entrenamiento de los algoritmos puede realizarse con el conjunto de datos completo del proveedor. Sin embargo, cuando surge la necesidad de evaluar los algoritmos, es necesario separar el conjunto de datos completo en dos subconjuntos, uno para entrenar los algoritmos y otro para evaluarlos. El proceso de Sampling o muestreo (Figura 6) provee un conjunto de datos para entrenamiento y otro para pruebas, necesarios para el entrenamiento y evaluación de los algoritmos que requieren ser priorizados. Para esto se presentan dos escenarios y dependen de la configuración del sistema para el proveedor:

1. Se define una configuración opcional que permite la realización de una prueba de significancia estadística en el proceso de priorización de

algoritmos, esto con el fin de tener una confianza sobre la priorización del algoritmo seleccionado. Si esta opción está activa, es necesario tomar *l* muestras independientes del conjunto de datos para realizar el proceso de entrenamiento y evaluación de cada algoritmo. Usar el método *k-fold* para obtener estas muestras, no asegura la independencia de éstas ya que cada *fold* es usado varias veces para el entrenamiento de los algoritmos.

2. Si la opción de la prueba de significancia estadística no está activa, se usa un método *k-fold* y por lo tanto cada algoritmo es entrenado y evaluado k veces.

El sistema permite determinar el número máximo de interacciones del conjunto de datos (*nmax*) que serán usadas para la priorización de los algoritmos. Si el tamaño del conjunto de datos es superior al número máximo de interacciones, se toman *nmax* interacciones de forma aleatoria y sobre estas se aplica el proceso de partición de datos para entrenamiento y evaluación de los algoritmos.

El uso de un conjunto de datos demasiado grande tiene un costo computacional más alto ya que las k particiones serán más grandes y cada entrenamiento y evaluación requerirán más tiempo y recursos. Por otro lado, no es necesario todo el conjunto de datos para priorizar los algoritmos.

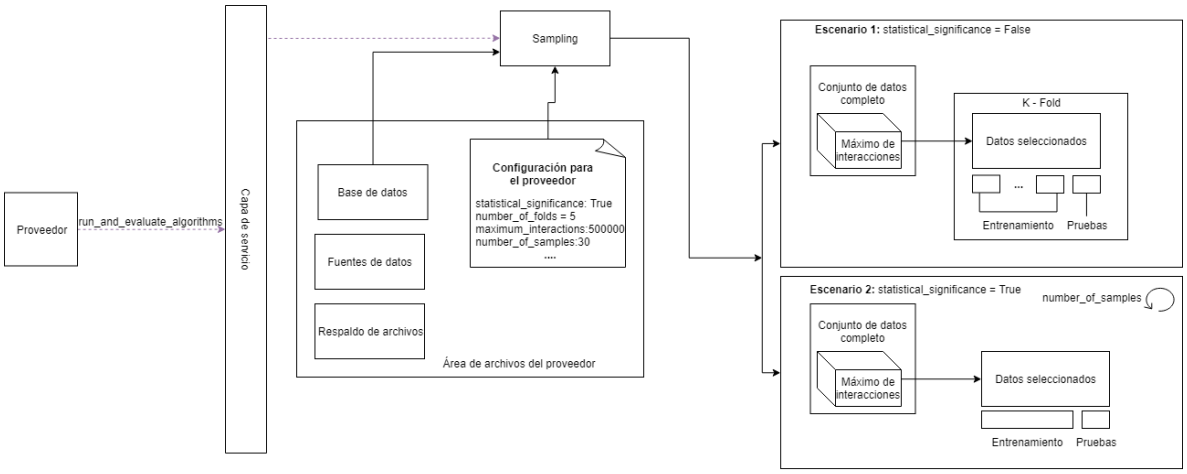

Figura 6:Proceso de Sampling o muestreo

El proceso de entrenamiento (Figura 7) se realiza para cada uno de los algoritmos del sistema y este hace uso del conjunto de datos de entrenamiento, resultante en cada iteración del proceso k-fold. El objetivo de este proceso es encontrar los valores que minimicen la función de costo definida en cada algoritmo, para ello emplea los valores definidos en el proceso de selección de hiperparámetros.

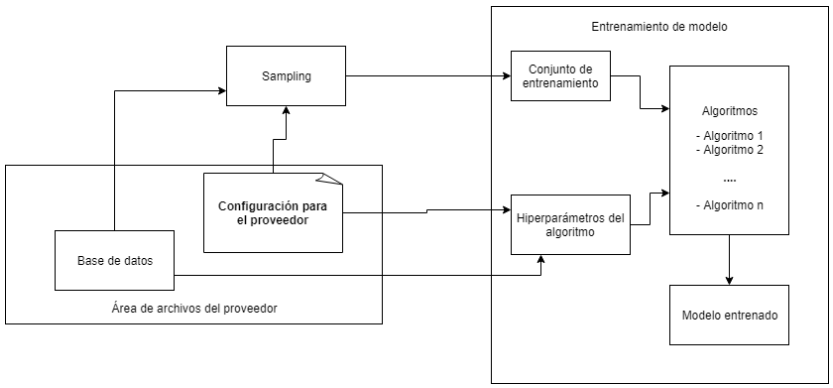

Figura 7: Proceso de entrenamiento de los algoritmos

El proceso de evaluación (figura 8) recibe el modelo entrenado y el conjunto de datos de prueba obtenido en la misma iteración del método *k-fold* en la que se

entrenó el modelo. Se pide al modelo que prediga la calificación para cada usuario-artículo en el conjunto de prueba, esta predicción se compara con la calificación real que dio el usuario al artículo y la diferencia entre ambos valores se usa para evaluar el algoritmo en métricas de exactitud de la predicción.

Para métricas basadas en el uso de las recomendaciones, se le pide al modelo que genere *m* recomendaciones para cada usuario en el conjunto de pruebas y sobre esas recomendaciones se analiza en qué proporción se encuentran, o no, los artículos que el usuario calificó de forma positiva en el conjunto de pruebas.

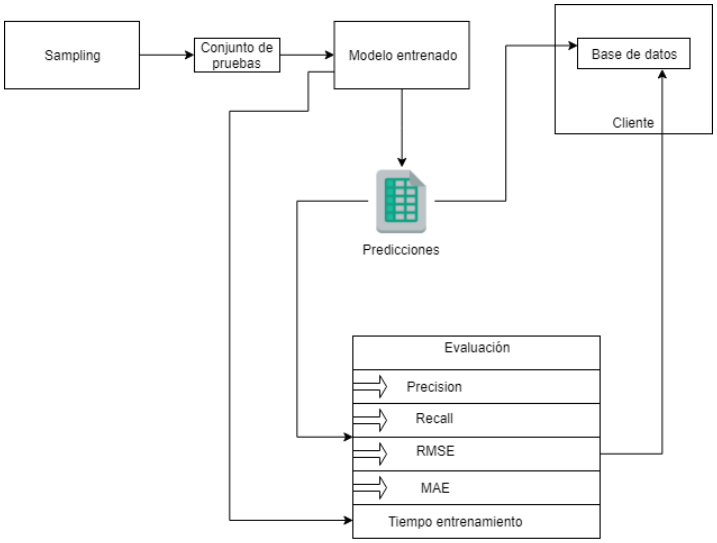

Figura 8: Proceso de evaluación de los modelos

En cada iteración del proceso *k-fold* se evalúan 5 métricas, dos de exactitud en la predicción de la calificación: RMSE y MAE, dos de uso de la recomendación: Precisión y Recall, y una de tiempo de entrenamiento del algoritmo. Los resultados de evaluación se almacenan en base de datos para cada iteración. Finalmente se promedian para encontrar el resultado general del algoritmo en cada métrica.

El proceso de priorización de algoritmos se encarga de tomar los valores de evaluación de todos los algoritmos y/o hibridaciones, los promedia y los usa con un modelo de combinación lineal (se define en la sección 3.2). Este modelo entrega un

valor total para cada algoritmo o hibridación, y de acuerdo con este valor, el listado de algoritmos se ordena de forma descendente, quedando en primer lugar el algoritmo con mejor resultado para los parámetros del modelo de priorización.

El proceso de hibridación de los algoritmos se presenta en la figura 9. Desde la base de datos se toman los resultados totales obtenidos por el modelo de priorización para cada algoritmo. En la configuración del sistema para el proveedor específico, se permite definir un umbral entre 0 y 1 que determina cuales algoritmos entran al proceso de hibridación, esto con el fin de limitar el número de algoritmos a hibridar, usando solo los que tienen mejor resultado y alivianando el costo computacional que conlleva cada hibridación. Todos los algoritmos cuyo valor total sea igual o mayor al valor del umbral, son tenidos en cuenta para la hibridación. También se puede definir el número máximo de algoritmos a hibridar y en caso de que ninguno de los algoritmos supere el umbral definido, solo se hibridan los dos primeros algoritmos con mejor resultado.

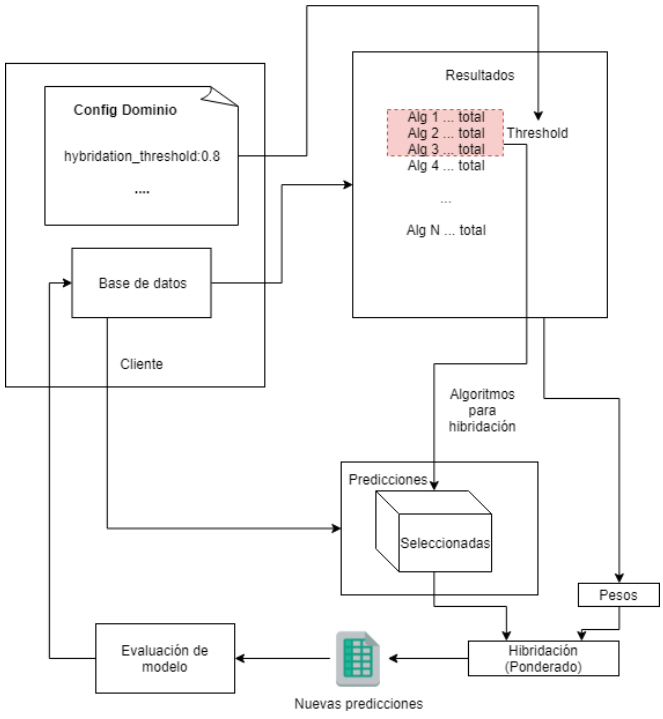

Figura 9: Proceso de hibridación de algoritmos

Luego de seleccionar los algoritmos a hibridar, se recupera desde la base de datos todas las predicciones para los algoritmos, en cada una de las iteraciones realizadas por el método *k-fold*. Las predicciones del algoritmo 1 y el algoritmo 2 se ponderan usando como peso el valor total de que tan bien lo hizo en las métricas de evaluación. Este proceso de hibridación genera nuevas predicciones, las cuales son evaluadas y sus resultados son almacenados en la base de datos para luego realizar una nueva priorización sobre el listado completo de algoritmos individuales y algoritmos hibridados.

El modelo de priorización de algoritmos incluye una prueba de significancia estadística que permite obtener un nivel de confianza frente a la posición de los algoritmos priorizados.

Por último, el sistema propuesto usa el algoritmo en el primer lugar de la lista, entregada por el proceso de priorización, para entrenarlo en el conjunto de datos completo y generar recomendaciones que serán consumidas por el proveedor en producción.

En la sección 3.3 se especifican los servicios que permiten interactuar con el sistema propuesto por medio de la capa de servicio.

MÉTODO DE ESTANDARIZACIÓN DE DATOS

Se propone un método de estandarización de datos (Figura 10) que permite recibir fuentes de datos con diferentes formatos y estructura.

Este método requiere que el proveedor entregue al sistema los archivos que contienen los datos para una fuente de información específica, y que defina la configuración de mapeo para esos archivos. Todo esto por medio de la capa de servicio del sistema.

La configuración de una fuente de datos contiene información relacionada con el formato y estructura del archivo, además de reglas de mapeo que identifican en los archivos del proveedor donde están los datos requeridos por los algoritmos del sistema.

El método de estandarización propuesto está diseñado para permitir la configuración, almacenamiento y proceso ETL para diferentes fuentes de información. Aun cuando en este trabajo solo de usaran datos explícitos con ratings.

Los archivos se almacenan en un área de archivos que corresponde a la fuente de datos indicada por el proveedor. Para esa fuente de datos debe existir un archivo de configuración que coincida con los archivos almacenados. Una vez iniciado el proceso de estandarización de datos, se revisa para cada fuente de información si tiene archivos almacenados y una configuración definida. En caso positivo, todos los archivos pasan al proceso ETL en el que son cargados en memoria con su estructura original, se les aplican reglas de transformación definidas en el archivo de configuración:

- De acuerdo con el mapeo de los parámetros en el archivo de configuración, se identifica cada parámetro en los datos de entrada y se transforman a un formato estándar para el sistema, usando en nombre del parámetro tal y como lo requieren los algoritmos en la etapa de entrenamiento.
- Se validan y eliminan datos vacíos, cuando es necesario.
- Se verifica que los valores en cada registro para el campo de calificación estén dentro del rango establecido en el archivo de configuración.
- Si está definido en la configuración, se retiran registros cuyo valor en ciertos campos corresponda al definido (limpieza de datos definida por el proveedor).

Si el archivo de datos es muy grande como para almacenarse en memoria, el sistema lo particiona en subconjuntos de datos, los procesa y luego une los resultados en base de datos.

Por último, los datos transformados se almacenan en base de datos para la fuente de información específica. Cada algoritmo de recomendación está definido para consumir datos de una o varias fuentes de información y este método de estandarización permite que los datos estén en el formato y con la estructura adecuada para el de entrenamiento de los algoritmos.

El sistema también puede recibir interacciones individuales (no mediante un archivo con varios registros). Estas también pasan por el proceso ETL según la fuente de información definida y su configuración.

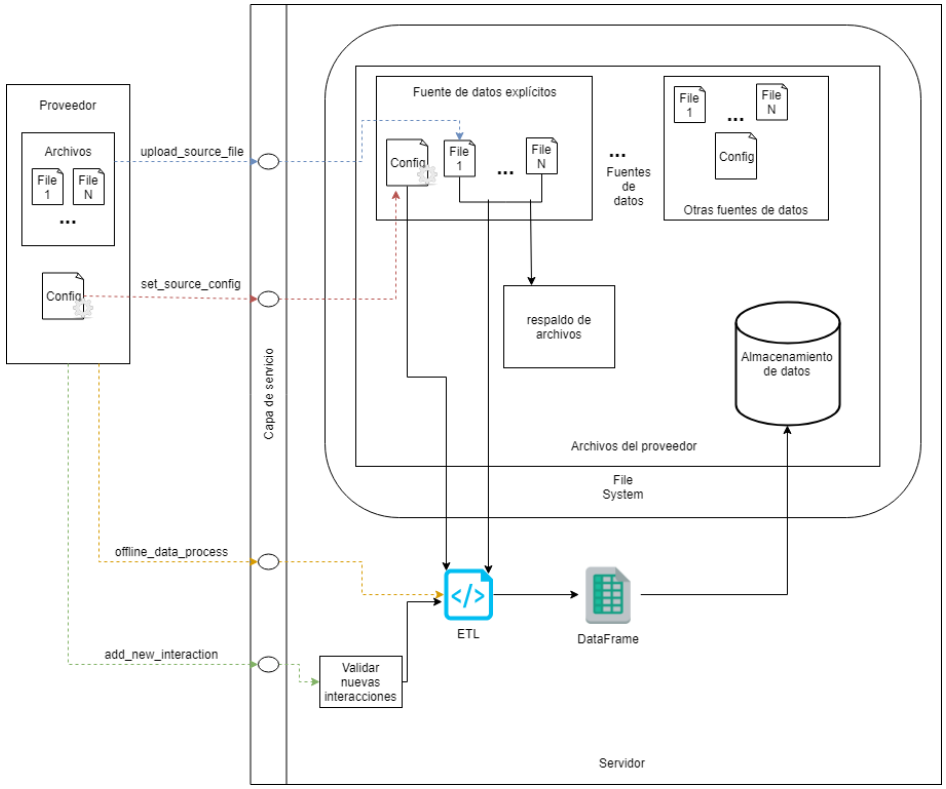

Figura 10: Proceso de extracción y estandarización de datos

MODELO DE PRIORIZACIÓN DE ALGORITMOS

Cada algoritmo es evaluado en 5 métricas diferentes:

- M1: RMSE
- M2: MAE
- M3: Precisión
- M4: Recall
- M5: Tiempo de entrenamiento

Los resultados de evaluación en cada una de estas métricas son tomados k veces para cada algoritmo y almacenados en base de datos. El valor de *k* puede ser igual a *l* dependiendo del proceso de muestreo aplicado: *k* para k-fold, *l* para número de muestras tomadas para la prueba de significancia estadística.

Los valores de las métricas deben normalizarse entre 0 y 1, indicando el grado en que mejor o peor lo hacen entre todas las muestras para todos los algoritmos, siendo 1 el mejor valor y 0 el peor valor. La normalización es necesaria ya que algunas métricas como RMSE y MAE pueden tomar valores mayores que 1 y el mejor valor es el que más se acerque a 0, por otro lado, las métricas Precisión y recall solo toman valores entre 0 y 1, siendo 1 el mejor valor. La métrica de tiempo también puede tomar valores muy altos, siendo mejores los valores más bajos.

Métricas normalizadas: nM1, nM2, nM3, nM4, nM5

El modelo usa un porcentaje de peso asignado a cada métrica (w1, w2, w3, w4, w5), el cual indica la influencia de la métrica en la priorización de los algoritmos. Este valor se define en la configuración del proveedor y por defecto se asigna el mismo porcentaje a todas las métricas.

Los pesos de cada métrica de evaluación permiten un mayor nivel de configuración y afinación del sistema, permitiendo hacer experimentos con estos valores. Pueden configurarse por medio de un servicio y se requiere un conocimiento del efecto de cada métrica y como afecta las necesidades del proveedor específico. En caso contrario, se recomienda mantener la configuración por defecto. Se propone para trabajos futuros la adaptación de un proceso automático que luego de priorizar los algoritmos con una distribución uniforme de los pesos de cada métrica (valor por

defecto), analice la diferencia significativa entre el mejor y el peor algoritmo y por cada métrica pueda sugerir el porcentaje adecuado.

Para hallar el total de cada algoritmo/hibridación A, en la iteración del muestreo k, el modelo multiplica el valor normalizado nM de cada métrica por el peso asignado a la misma w.

$$total_{A,k} = \left(nM1_{A,k} * w1\right) + \left(nM2_{A,k} * w2\right) + \left(nM3_{A,k} * w3\right) + \left(nM4_{A,k} * w4\right) + \left(nM5_{A,k} * w5\right)$$

Para cada algoritmo A se promedian todos los totales de las iteraciones k y con esto se halla el valor total para el algoritmo que indica que tan bien lo hace en general.

$$Total_A = \overline{\chi}\left(total_{A,1}, total_{A,2}, \ldots, total_{A,k}\right)$$

Por último, se ordenan los algoritmos en forma descendente con respecto al valor de $Total_A$, consiguiendo así la priorización de estos. El algoritmo en el primer lugar es el que mejor lo hace para el conjunto de datos en las métricas definidas, teniendo en cuenta los pesos asignados a cada métrica. Este resultado se almacena en la base de datos como el seleccionado para proporcionar las recomendaciones al proveedor específico.

El algoritmo seleccionado debe entrenarse con el conjunto de datos completo para comenzar a entregar recomendaciones en producción.

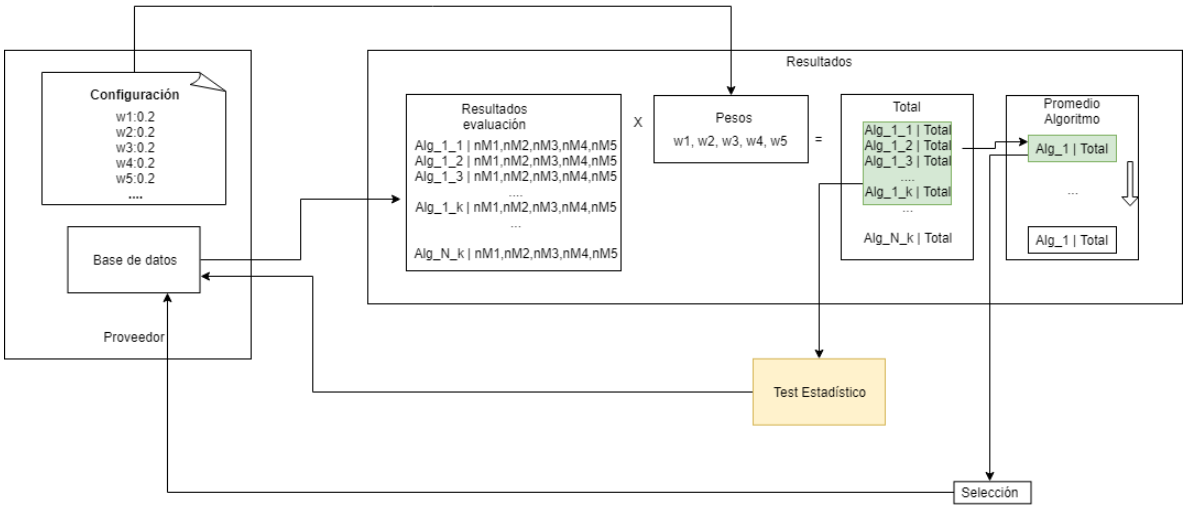

Figura 11: Proceso de priorización y selección de algoritmos

Significancia estadística: La prueba estadística permite determinar si al comparar las métricas de evaluación entre dos algoritmos, la diferencia entre estos es real o es el resultado de una casualidad estadística.

Las pruebas de significancia estadística mejoran la confianza en la interpretación y la presentación de los resultados durante la selección del algoritmo.

El prototipo incluye un módulo de significancia estadística que se usa principalmente para el análisis en la selección del modelo durante las pruebas del prototipo. Este se puede activar y desactivar desde la configuración del cliente y según su estado (activo/inactivo) se cambia el comportamiento al momento de tomar las muestras del conjunto de datos para entrenamiento y pruebas.

Desde un punto de vista del cliente, este puede activar la prueba estadística para obtener información de la confianza que se tiene en que el modelo seleccionado seguirá siendo el mismo para otras muestras de la misma población.

La prueba estadística t-test tiene como hipótesis nula H0 = No hay significancia estadística en la diferencia entre las muestras. El valor de p-value indica la

probabilidad de que se acepte la hipótesis nula y por lo tanto 1 - p-value es la probabilidad de que se rechace la hipótesis nula.

DESCRIPCIÓN DE SERVICIOS

Los servicios web para el consumo del sistema no son implementados en el desarrollo del prototipo. Sin embargo, son considerados como una parte fundamental en la propuesta de la solución y por lo tanto se presenta una especificación básica de los mismos.

Gestión de archivos

- Archivos de interacciones: por medio de una interfaz web, debe existir una comunicación HTTP que permita al cliente subir archivos (uno o muchos) al servidor y que estos se almacenen en el directorio del cliente sobre los archivos por procesar.
- Eliminar archivos de interacciones: por medio de una interfaz web, se debe permitir eliminar archivos que se encuentran en el directorio de archivos por procesar para el cliente.

Gestión de la configuración del sistema

- Configuración de cliente: Por medio de una interfaz web, el cliente puede cambiar los valores de configuración, los cuales alteran el comportamiento de todas las etapas del sistema.
- Configuración de fuente: Por medio de una interfaz web, el cliente debe configurar las fuentes de información a usar, dependiendo del formato de los archivos que haya subido al servidor en cada fuente de información.

Gestión de procesos

- Obtener datos – ETL para fuentes de datos (incremental): inicia el proceso de validar los archivos de interacciones del cliente para todas las fuentes de información, según la configuración presentada por el usuario. Una vez validados, se concatenan todos los archivos, se extraen, transforman y

cargan los datos en la base de datos del cliente. Este proceso se puede ejecutar múltiples veces y sólo alimentará la base de datos con nuevas interacciones, manteniendo las anteriores extraídas desde otras ejecuciones.

- Obtener datos – ETL para fuentes de datos (total): Igual al anterior pero siempre que se ejecuta vuelve a procesar todos los archivos, incluyendo los que ya se procesaron en una ejecución anterior. Este proceso borra los datos cargados en la base de datos antes de iniciar, y carga los nuevos datos de su resultado.
- Ejecutar y evaluar algoritmos: Este proceso toma el conjunto de datos, toma las muestras para entrenamiento y prueba, entrena y evalúa los algoritmos.
- Hibridar algoritmos: toma los resultados de evaluación de algoritmos individuales y los híbrida según lo definido en la configuración del cliente.
- Seleccionar algoritmo (nuevo o continuo): Si es nuevo, ejecuta los procesos ejecutar y evaluar algoritmos e hibridar algoritmos, finalmente selecciona el mejor algoritmo según la configuración del cliente. Si es continuo y existen datos de los procesos anteriores, solo selecciona el algoritmo con mejor resultado.
- Obtener significancia estadística entre algoritmos: Presenta los resultados de la prueba estadística para dos algoritmos ingresados como parámetros al servicio.
- Entrenar y evaluar seleccionado: Permite entrenar y evaluar solo el algoritmo seleccionado.

Enviar nuevas interacciones al conjunto de datos del proveedor

- User-item-rating: Permite enviar por medio de un servicio web, nuevas interacciones hacia una fuente de información del sistema. Estas alimentarán el conjunto de datos y servirán para un próximo entrenamiento de los algoritmos.

Servicio de recomendación:

Estas recomendaciones se pueden pedir desde una aplicación del cliente como un servicio web o desde una interfaz web de administración del cliente.

- Recomendar k artículo al usuario: El algoritmo seleccionado genera k recomendaciones para un usuario específico.
- Predecir rating para artículo por un usuario: El algoritmo seleccionado predice el rating que dará el usuario al artículo.
- Recomendar usuarios a un artículo: Recomienda usuarios que podrían estar interesados en el artículo.
- Recomendar usuarios a un usuario: Recomienda usuarios similares al usuario ingresado basándose en las interacciones pasadas del usuario.
- Recomendar artículos a un artículo: Recomienda conjuntos de ítems que de alguna forma están relacionados con el artículo ingresado
- Recomendar k artículos populares: Usando un algoritmo de popularidad, se recomiendan los k artículos más populares. Esto es útil cuando un usuario tiene ninguna o pocas interacciones.

Administrar artículos y usuarios

- Excluir artículo de todas las recomendaciones.
- Excluir artículo en las recomendaciones de un usuario específico.
- Eliminar artículo del conjunto de datos.
- Eliminar usuario del conjunto de datos.
- Eliminar artículo para usuario en las interacciones.
- Listar artículo.
- Listar usuarios.

IMPLEMENTACIÓN DEL PROTOTIPO

Tecnologías seleccionadas para prototipo

Se usa *Python* como lenguaje de desarrollo para el prototipo. *Python* tiene una sintaxis consistente y es fácil de leer y entender su código. Existen muchas librerías para Python que facilitan la manipulación de datos y el procesamiento matemático que es requerido al momento de implementar algoritmos de aprendizaje de máquina.

Se usa *Pandas* para *Python* como herramienta para la preparación y manipulación de datos. Esta permite trabajar con diferentes estructuras de datos (Series: una dimensión, Dataframe: dos dimensiones) de forma rápida, flexible y expresiva. Cuenta con múltiples funciones que facilitan tareas como el manejo de datos faltantes, mutabilidad de los datos (insertar y eliminar), organizar los datos por agrupamiento, fusionar y unir conjuntos de datos, entre otros.

NumPy es un paquete fundamental en *Python* para la computación científica. Este permite trabajar con vectores de muchas dimensiones con alto rendimiento, usar funciones de álgebra lineal y trabajar con números aleatorios.

SQLAlchemy es un kit de herramientas de base de datos que proporciona una interfaz estándar para crear y ejecutar código independiente de la base de datos sin necesidad de escribir sentencias de SQL.

Surprise es un paquete en *Python*, desarrollado independientemente para *SciPy*, que permite construir y analizar sistemas de recomendación. Proporciona herramientas para evaluar, analizar y comparar el rendimiento de los algoritmos. Ejecuta procedimientos de validación cruzada (cross validation) con potentes iteradores (inspirados en las herramientas de scikit-learn). También incluye herramientas para la búsqueda exhaustiva sobre un conjunto de parámetros. Este

paquete proporciona los algoritmos usados en el desarrollo de este prototipo y la función para encontrar los parámetros óptimos de cada algoritmo.

Proceso ETL

El proceso de extracción, transformación y carga de datos (ETL) se define de la siguiente forma:

1. Se dispone en el sistema de archivos del servidor, un directorio por cada cliente, este directorio contiene a su vez 3 subdirectorios que corresponden a fuentes de datos, respaldo de archivos y base de datos.

2. El directorio fuente de datos contiene subdirectorios que corresponden a diferentes fuentes de información: contenido de artículos, Explicito, Implícito, Explícito con comentarios, contenido de usuarios. En este prototipo solo se usa los datos explícitos, pero la extracción de datos se diseñó para extraer datos de diferentes fuentes de información y luego usarlos con algoritmos que reciban esos datos.

3. El directorio respaldo de archivos contiene los archivos del cliente, categorizados por fuente de información, que ya han sido procesados por el módulo ETL.

4. El directorio base de datos contiene la base de datos del cliente y otros archivos como predicciones de los algoritmos para hibridación.

5. El módulo ETL primero valida los directorios de fuentes de datos que contengan archivos por procesar y que tengan un archivo de configuración válido. Se toma la información del archivo de configuración y se extraen los datos de cada archivo, mapeándolos a los tipos requeridos por los algoritmos del sistema; este mapeo se obtiene directamente del archivo de configuración. Se concatenan los datos de todos los archivos en un solo *DataFrame* de la librería *Pandas* y se realizan transformaciones adicionales especificadas por el cliente en el archivo de configuración, por ejemplo: limpieza de los datos. Finalmente se almacena el *DataFrame* procesado en la base de datos del cliente.

6. Una vez termina el proceso de ETL sobre los archivos de las fuentes de datos, estos son movidos al directorio de respaldo de archivos, esto con el objetivo de no volver a procesarlos nuevamente y dejar libre el directorio de fuente de datos para

futuros archivos. En caso de pérdida de base de datos, se pueden volver a procesar todos los archivos desde el respaldo de archivos.

7. El sistema debe contar con un servicio que permita recibir, transformar y almacenar interacciones individuales para ciertas fuentes de información.

Selección de hiperparámetros

Los hiper-parámetros son parámetros que no son aprendidos directamente en el entrenamiento del modelo. Por lo tanto, estos parámetros deben ser pasados como argumentos a cada algoritmo.

Por medio de la librería Surprise de Python se usa la clase GridSearchCV la cual calcula métricas de exactitud de la predicción para un algoritmo en varias combinaciones de parámetros, a través de un procedimiento de validación cruzada (cross validation). Esto es útil para encontrar el mejor conjunto de parámetros para un algoritmo predictivo.

Entrenamiento del modelo

La librería *Surprise* para *Python* provee la implementación de algoritmos populares de filtro colaborativo los cuales representan una base interesante para el desarrollo y evaluación de este prototipo. Estos algoritmos se describen en el capítulo 2.4 de este documento. La librería se encarga del proceso de entrenamiento para cada uno de los algoritmos y el sistema propuesto se encarga de proporcionar todos los datos requeridos para el funcionamiento del algoritmo y su evaluación.

CAPITULO 4. VALIDACIÓN Y RESULTADOS

Con el prototipo desarrollado se busca validar la solución de diseño propuesta, esperando que este permita el uso de diferentes conjuntos de datos en el proceso de entrenamiento, hibridación y evaluación de algoritmos de recomendación de la técnica filtro colaborativo. Para luego, priorizar los algoritmos y seleccionar el que presente mejores resultados en las métricas propuestas.

Se probará con tres conjuntos de datos, usando diferentes dominios y formatos en los datos de entrada. Se presentará una descripción y análisis de los datos de entrada, su configuración, fuente de información a la que pertenecen y cómo se transforman para ser usados por los algoritmos del sistema. A continuación, los datos transformados serán usados para entrenar y evaluar los diferentes algoritmos, este proceso reportará las métricas de cada algoritmo en el conjunto de datos. Con estas métricas se halla un valor total por algoritmo teniendo en cuenta la importancia de cada una para el proveedor específico (pesos asignados en configuración). Estos totales servirán para definir los algoritmos a hibridar, los cuales serán evaluados en las mismas métricas. Finalmente se priorizan los algoritmos y se selecciona el primero de la lista, indicando que es el que mejor lo hace para las métricas definidas y los pesos asignados.

La selección del algoritmo irá acompañada por una prueba estadística *t-test* que indicará si la priorización es significativa o si ocurrió por el azar. Esto permitirá conocer la confianza sobre la posición del algoritmo seleccionado para la población total.

INFRAESTRUCTURA PARA PRUEBAS

Las pruebas del prototipo se realizan sobre un servidor Ubuntu versión 16.04.5 LTS. Con los siguientes recursos:

Procesador:

Nombre: Intel(R) Xeon® CPU ES-2699 v3 @ 2.30GHz

Cores: 72

CPU max MHz: 3600

Memoria: 520 GB

Capacidad almacenamiento: 520 GB

El proceso de prueba del prototipo solo trabaja con un procesador a la vez debido al no uso de procesos en paralelo, esto se propone para trabajos futuros. El consumo de memoria es reducido en todos los algoritmos basados en modelo. Sin embargo, para algoritmos basados en memoria se usa entre un 10% y un 15% de la memoria del servidor lo que corresponde a 78 GB aproximadamente.

CASOS DE ESTUDIO

El sistema de recomendación propuesto busca poder usar diferentes conjuntos de datos, con diferentes formatos y estructura. Esto con el objetivo de servir a diferentes proveedores en diversos dominios. Para ello el sistema incluye un método de estandarización de los datos y un modelo de priorización de algoritmos, además de los procesos comunes en un sistema de recomendación: entrenamiento, hibridación y evaluación de algoritmos.

Para probar el prototipo desarrollado y validar que efectivamente pueda ser usado por diferentes proveedores, es necesario evaluarlo con diferentes conjuntos de datos, con diferentes propiedades, diferentes formatos: CSV con encabezados, sin encabezados y JSON, y diferentes dominios: películas, productos, lugares.

Se seleccionan y usan solo datos explícitos para la evaluación del prototipo. El motivo de esta decisión reside en su popularidad y en la disponibilidad de los conjuntos de datos explícitos para pruebas. Estos son compartidos en Internet por la comunidad que investiga sobre sistemas de recomendación, los trabajos encontrados en el estado del arte trabajan principalmente con datos explícitos y algoritmos de filtro colaborativo. Específicamente, los conjuntos de datos usados en este trabajo son bastante populares para la investigación en sistemas de recomendación.

El sistema propuesto incluye un modo para obtener la significancia estadística de los resultados de priorización de los algoritmos. Este modo cambia la forma como se toman las muestras del conjunto de datos para los procesos de entrenamiento y pruebas. Por lo tanto, se considera importante hacer uso de este modo en la evaluación del prototipo y también evaluar el comportamiento cuando este modo no se usa. De acuerdo con lo anterior se define hallar la significancia estadística en solo dos de los conjuntos de datos a usar.

Se debe tener en cuenta que el modo de significancia estadística requiere un costo computacional más alto, ya que debe entrenar y evaluar los algoritmos en un número superior de muestras. Se hace uso de este modo, principalmente para la evaluación del prototipo, pero queda definido el proceso para el uso opcional por los usuarios del sistema.

El proceso de priorización de algoritmos requiere los valores de los pesos para cada una de las métricas evaluadas. Se recomienda dejar los valores por defecto para este parámetro a menos que se conozcan las fortalezas de cada métrica y se alineen con los intereses de cada proveedor específico. Los valores por defecto se

encuentran uniformemente distribuidos, otorgándole el mismo peso a cada una de las métricas. En la evaluación del prototipo, se busca probar el uso de este en diferentes escenarios, con diferentes conjuntos de datos. Por otro lado, no es objetivo de este trabajo el comparar los resultados obtenidos por los tres conjuntos de datos a utilizar. De acuerdo con lo anterior, se variarán los valores en cada caso, dejando uno de ellos con los valores por defecto.

El modelo de hibridación de algoritmos hace uso de dos propiedades en la configuración del sistema, ambas propiedades determinan los algoritmos que serán hibridados desde la lista de algoritmos priorizados. El primer valor (hybridation_threshold) corresponde al umbral o valor mínimo requerido en el total de la priorización, para decidir si el algoritmo será hibridado o no. El segundo valor (max_for_hybrid) corresponde al número máximo de algoritmos a hibridar. Estos valores permiten limitar la cantidad de algoritmos en el proceso de hibridación, reduciendo así el costo computacional.

Por defecto se definen los siguientes valores: hybridation_threshold=0.8 y max_for_hybrid=4, esto con el fin de reducir el costo computacional de tener que hibridar todos los algoritmos del sistema, y seleccionar solo los primeros 4 algoritmos que lo hayan hecho al menos un 80% bien en las métricas de evaluación relevantes para el modelo de priorización.

Con el objetivo de aplicar diferentes valores en la evaluación del prototipo, se cambiarán estos parámetros en cada uno de los conjuntos de datos, a excepción de Yelp que mantendrá los valores por defecto.

Movielens

Descripción: Movielens es un sistema de recomendación web que recomienda películas a los usuarios basandose en sus preferencias y en técnicas como filtro colaborativo. Movielens fue creado en 1997 por el grupo de investigación GroupLens. Movielens pone a disposición de la comunidad conjuntos de datos de

diferentes tamaños y con diferentes características. Estos son usados generalmente para la investigación, en especial sobre sistemas de recomendación.

Dominio: Películas.

Conjunto de datos:

- Nombre archivo: movies.csv
- Número de interacciones: 20000263
- Tamaño del conjunto de datos: 508 MB
- Busca recomendar: Películas.
- Información incluida(columnas): userId,movieId,rating,timestamp
- Formato del archivo: Los archivos del conjunto de datos están codificados en UTF-8, son archivos separados por comas y contienen una fila con las cabeceras de los datos. Estos archivos se encuentran en formato CSV.
- Filas: 20000263
- Columnas: 4
- Tipo de rating: explicito – númerico – decimal
- Rango de rating: (0.5 – 5)

- Configuración del cliente

```
[EVALUATION]
precision_weight = 0
recall_weight = 0.5
time_weight = 0.1
rmse_weight = 0.4
mae_weight = 0

[RESULTS]
statistical_significance = true
number_of_2_fold_samples = 20
min_population_constraint = 50000
test_metric = total

[CLEANING]
min_interactions_explicit = 4

[SAMPLING]
number_of_folds = 1
test_size = 0.25
explicit_rating_threshold_for_test = 4
maximum_interactions_evaluation = 500000

[RECOMMEND]
recommendations_per_user=20

[HYBRIDATION]
hybridation_threshold = 0.6
max_for_hybrid = 4
```

Para este cliente se activa la prueba de significancia estadística (statistical_significance), lo que permitirá conocer la confianza en la priorización de los algoritmos. De acuerdo a lo anterior, en la etapa de sampling no se usará la técnica K-fold sino que se tomarán varias muestras de la población y sobre cada una de estas se tomará un 25% para pruebas y un 75% para entrenamiento, como indica la propiedad (test_size). La propiedad (explicit_rating_threshold_for_test) indica el valor de rating mínimo que se debe tomar para el conjunto de pruebas y la

propiedad (maximum_interactions_evaluation) indica el tamaño máximo de cada muestra.

En Cleaning se indica que solo se tendrán en cuenta usuarios con al menos 4 interacciones para el entrenamiento de los algoritmos. Esta decisión se debe al conocimiento que los algoritmos de filtro colaborativo funcionan mejor para usuarios con al menos 3 interacciones.

Se generan listas de recomendaciones de 20 items para cada usuario en el conjunto de pruebas como indica la propiedad (recommendations_per_user). Estas listas permiten evaluar las métricas de uso de la recomendación y generar los listados de recomendaciones en producción.

Para la hibridación se tomarán los algoritmos con un valor total superior a 0.6 y se tomarán como máximo 4 algoritmos.

En la evaluación de los algoritmos se define un interés en las métricas RMSE en un 40%, Recall en un 50% y tiempo de entrenamiento en un 10%, indicando un mayor interés en el uso de las recomendaciones, pero no muy alejado de la importancia para la exactitud de las predicciones. Por otro lado, el tiempo de entrenamiento se considera solo un poco importante.

Proceso ETL:

- Configuración para el mapeo (ETL): Este archivo contiene la configuración definida para el proveedor y que permite representar el formato original de los datos del cliente hacia el formato usado por el sistema.

```
{
    "input_type":"explicit",
    "is_json": false,
    "file_type_information”: {
```

```
        "csv_headers": true,
        "csv_separator":","
    },
    "key_map": {
        "rating":"rating",
        "user_id":"userId",
        "item_id":"movieId",
        "timestamp":"timestamp",
        "event_side_features":"",
        "rating_config": {
            "min_rating":0.5,
            "max_rating":5,
        },
        "remove_content":"",
    }
```

- Reglas de limpieza aplicadas: Se retiran los usuarios con menos de 4 interacciones para la selección de algoritmos de filtro colaborativo, ya que este tipo de algoritmos no responden bien para usuarios con pocas interacciones. Al entrenar el algoritmo seleccionado se deben tener en cuenta todos los usuarios y recomendar los artículos más populares a los usuarios con pocas interacciones.
- Formato resultante y caracteristicas de los datos:

index	target	user_id	timestamp	item_id
0	3.5	1	1112486027	2
1	3.5	1	1112484676	29
2	3.5	1	1112484819	32
3	3.5	1	1112484727	47
4	3.5	1	1112484580	50
5	3.5	1	1094785740	112
6	4.0	1	1094785734	151
7	4.0	1	1112485573	223
8	4.0	1	1112484940	253
9	4.0	1	1112484826	260
10	4.0	1	1112484703	293

Figura 12: Dataframe resultante - Movielens

Luego de ejecutar el proceso de ETL se obtiene un Dataframe de Pandas (figura 12). Este corresponde al formato general usado por los algoritmos de recomendación del sistema.

- Número de usuarios: 138493 usuarios.
- Número de artículos: 26744 artículos.
- Número de ratings: 20000263
- Porcentaje de valores vacíos en la matriz: 0.9946001521864456
- Descripción estadística del conjunto de datos:

	target
count	2.000026e+07
mean	3.525529e+00
std	1.051989e+00
min	5.000000e-01
25%	3.000000e+00
50%	3.500000e+00
75%	4.000000e+00
max	5.000000e+00

Tabla 2: Descripción estadística - Movielens

- Distribución de los ratings en el conjunto de datos:

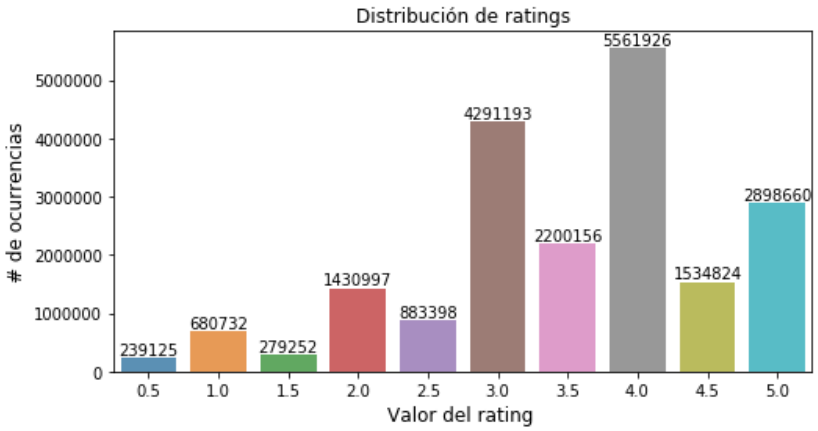

Figura 13: Distribución de ratings - Movielens

La información anterior nos indica que hay muchos más usuarios que artículos. Por lo tanto es más óptimo computacionalmente para las técnicas de filtrado colaborativo basadas en memoria, usar un enfoque orientado en artículo, ya que un enfoque basado en usuario va a requerir cargar una matriz más grande en memoria. La matriz de interacciones está conformada en un 99% por valores vacíos, lo cual es común en este tipo de conjuntos de datos. Aun así, de acuerdo a la densidad de la matriz algunos algoritmos podrían presentar mejores resultados.

En la tabla 2 y en la figura 13 se muestra la distribución de las calificaciones, la media y la desviación estándar. Según estos datos se puede ver que la mayoría de calificaciones toman valores entre 3 y 5, la media es 3.52 y la desviación es de 1.05. esto indica que los usuarios del conjunto tienden a dar calificaciones altas a los artículos y por lo tanto hay más información que le permita a los algoritmos aprender sobre lo que le gusta a los usuarios pero no hay tanta información que represente lo que no les gusta, lo cual puede sesgar al modelo dificultando el distinguir lo que no le gusta al usuario.

Proceso de entrenamiento y evaluación de algoritmos:

El conjunto de datos obtenido en el paso anterior es usado para entrenar y evaluar los algoritmos del sistema. Los resultados (sin normalizar) para cada algoritmo en las diferentes métricas son los siguientes:

model	mae	rmse	precision	recall	time
surprise_BaselineOnly	0.709721603	0.91614407	0.944645229	0.496132783	1.540284491
surprise_svd	0.712868997	0.920174218	0.9303363	0.519589213	18.79095618
surprise_KNNBaseline	0.775200306	1.015565015	0.894594569	0.569908036	6.042401958
surprise_KNNWithMeans	0.79153984	1.031588253	0.895516223	0.553053716	4.764174247
surprise_KNNWithZScore	0.791747006	1.033506998	0.893679515	0.555391652	5.328160417
surprise_SVDpp	0.710516504	0.917492013	0.933874099	0.516533086	64.06918536
surprise_CoClustering	0.811195291	1.047458442	0.912552805	0.548901403	14.27238319
surprise_SlopeOne	0.885844275	1.151100456	0.85695505	0.577466857	5.477397013
surprise_KNNBasic	0.863979024	1.12935408	0.856612147	0.539051235	4.630370402
surprise_NMF	0.831223486	1.0690986	0.931419565	0.484699741	25.35242692
surprise_NormalPredictor	1.144677595	1.435627742	0.769626512	0.544874642	0.372364831

Teniendo en cuenta los pesos definidos en la configuración del cliente y luego de normalizar los valores de cada métrica, se aplica el modelo de priorización y se obtienen los siguientes resultados totales para los algoritmos independientes:

model	norm_mae	norm_rmse	norm_time	total
surprise_BaselineOnly	1	1	0.981664396	0.746232831
surprise_svd	0.992763881	0.992242011	0.710839706	0.727775381
surprise_KNNBaseline	0.849459017	0.808615842	0.910983985	0.699498753
surprise_KNNWithMeans	0.81189307	0.777771296	0.931051356	0.680740512
surprise_KNNWithZScore	0.811416777	0.774077734	0.922197128	0.679546633
surprise_SVDpp	0.998172457	0.997405226	0	0.657228633
surprise_CoClustering	0.766703552	0.747221367	0.781778459	0.651517094
surprise_SlopeOne	0.59507933	0.547711701	0.919854207	0.59980353
surprise_KNNBasic	0.645349361	0.589573221	0.933151992	0.598670105
surprise_NMF	0.720657066	0.705564317	0.607828744	0.585358471
surprise_NormalPredictor	0	0	1	0.372437321

Cada métrica debe aportar en un porcentaje al valor total que permite priorizar los algoritmos. Algunos valores como precisión y recall son mejores mientras están más

cerca de 1. Por otro lado, valores como MAE, RMSE y time son mejores mientras estén más cerca de 0. Es por esto que MAE, RMSE y time son normalizados a valores entre 0 y 1, y su valor original es invertido para que el aporte al total sea correspondiente a su cercanía a 0.

Proceso de hibridación de algoritmos:

Según la configuración del cliente, para la hibridación se tomarán como máximo 4 algoritmos que tengan un valor total por encima de 0.6, suponiendo que para el proveedor son relevantes para hibridación todos los algoritmos que lo hagan al menos un 60% bien en las métricas de evaluación. De acuerdo con esto se seleccionan los siguientes algoritmos:

['surprise_BaselineOnly', 'surprise_svd', 'surprise_KNNBaseline', 'surprise_KNNWithMeans']

La hibridación y evaluación de los algoritmos seleccionados presenta los siguientes resultados (sin normalizar):

model 1	model 2	mae	precision	recall	rmse	time
surprise_BaselineOnly	surprise_KNNBaseline	0.71624108	0.92722025	0.53368839	0.92703072	7.58268645
surprise_BaselineOnly	surprise_KNNWithMeans	0.7191019	0.93239125	0.5130132	0.92802613	6.30445874
surprise_BaselineOnly	surprise_svd	0.708292	0.93978508	0.50683721	0.91461034	20.3312407
surprise_svd	surprise_KNNBaseline	0.71856968	0.92149811	0.54435709	0.93040754	24.8333581
surprise_svd	surprise_KNNWithMeans	0.72174352	0.92739449	0.52155079	0.93165281	23.5551304
surprise_KNNBaseline	surprise_KNNWithMeans	0.77903085	0.90101769	0.5472678	1.01716616	10.8065762

Teniendo en cuenta los pesos definidos en la configuración del cliente y luego de normalizar los valores de cada métrica, se aplica el modelo de priorización y se obtienen los siguientes resultados totales para las hibridaciones:

model 1	model 2	norm_mae	norm_rmse	norm_time	total
surprise_BaselineOnly	surprise_KNNBaseline	0.88762774	0.87889147	0.93101437	0.71150222
surprise_BaselineOnly	surprise_KNNWithMeans	0.84718585	0.86918543	1	0.70418077
surprise_BaselineOnly	surprise_svd	1	1	0.24297814	0.67771642

surprise_svd	surprise_KNNBaseline	0.85470948	0.84596491	0	0.61056451
surprise_svd	surprise_KNNWithMeans	0.80984255	0.83382247	0.06898563	0.60120295
surprise_KNNBaseline	surprise_KNNWithMeans	0	0	0.75702186	0.34933609

Priorización y selección de algoritmos:

model	norm_mae	norm_rmse	norm_time	total
surprise_BaselineOnly	0.99672399	0.99705628	0.95324593	0.7422135
surprise_BaselineOnly_surprise_KNNBaseline	0.98178426	0.97616129	0.71135694	0.7284444
surprise_BaselineOnly_surprise_KNNWithMeans	0.97522856	0.97425077	0.76252686	0.72245959
surprise_KNNBaseline	0.84667618	0.8062355	0.77301749	0.68474997
surprise_svd	0.98951157	0.98932113	0.26266831	0.68178989
surprise_BaselineOnly_surprise_svd	1	1	0.20100776	0.67351938
surprise_KNNWithMeans	0.8092333	0.77548175	0.82418741	0.6691383
surprise_KNNWithZScore	0.80875857	0.77179907	0.80160996	0.66657645
surprise_svd_surprise_KNNBaseline	0.97644816	0.9696801	0.02077932	0.66212852
surprise_svd_surprise_KNNWithMeans	0.96917515	0.96729001	0.07194924	0.65488633
surprise_KNNBaseline_surprise_KNNWithMeans	0.83789829	0.80316239	0.58229842	0.6531287
surprise_CoClustering	0.76419182	0.74502176	0.44355549	0.61681495
surprise_KNNBasic	0.64323519	0.58783768	0.82954384	0.58761507
surprise_SlopeOne	0.59312984	0.54609939	0.79563573	0.58673676
surprise_NMF	0.71829618	0.70348733	0	0.5237448
surprise_NormalPredictor	0	0	1	0.37243732

La normalización de las métricas depende de los valores de los otros algoritmos con los que se compara para la priorización. Es por esto, que al agregar las hibridaciones a los resultados y aplicar de nuevo el modelo de priorización, los valores totales cambian con respecto a los totales de los algoritmos individuales o hibridaciones por separado.

Prueba estadística t-test:

El sistema realiza la prueba estadística sobre una métrica definida por el usuario, para este cliente se usa la métrica: total, lo que indica que sobre todas las muestras tomadas por cada algoritmos, se usará el valor de total para la prueba t-test.

Se realiza la prueba estadística sobre las tres primeras posiciones de los resultados para conocer la probabilidad de que la primera posición sea reemplazada por alguna de las dos siguientes. A continuación se presenta el resultado:

Muestras para total:

Posición 1: surprise_BaselineOnly

[0.73997358 0.7402207 0.74015254 0.73988151 0.73946224 0.73747161

 0.74085258 0.74062761 0.73851134 0.73880164 0.74041375 0.74120111

 0.74086211 0.74056486 0.73944453 0.73872332 0.74138188 0.73860163

 0.7383914 0.74330146]

Posición 2: surprise_BaselineOnly_surprise_KNNBaseline

 [0.72677238 0.72632989 0.7281089 0.72779415 0.72742453 0.72539798

 0.72758655 0.72679646 0.72618611 0.72580442 0.72684404 0.72839592

 0.72824288 0.72711141 0.72621892 0.72628204 0.72825689 0.72620208

 0.7270832 0.73091009]

Posición 3: surprise_BaselineOnly_surprise_KNNWithMeans

[0.72030015 0.7197088 0.72155186 0.72100117 0.72122011 0.71991195

 0.72158589 0.72073787 0.71977563 0.72078334 0.72184136 0.7220067

 0.72170624 0.72083535 0.72012255 0.71975896 0.72181467 0.72009565

 0.72107468 0.7245846]

Resultado:

Posición 1 – Posición 2:

statistic=31.452017660249357

pvalue=8.150182163158312e-29

Posición 2 – Posición 3:

statistic=-16.42118469653899

pvalue=7.567738689681451e-19

El valor de p-value para ambos casos es menor que 0.05 por lo tanto se rechaza la hipótesis nula y se concluye que existe significancia estadística en la diferencia de las muestras. Se asume que para la población se mantendrán los mismos algoritmos en la posición 1 y la posición 2.

En este caso, se evaluó el prototipo con un conjunto de datos relacionado con el dominio de películas y en donde los datos de entrada se encontraban en archivos con formato CSV separado por comas y con encabezados. El sistema transformó correctamente los datos de entrada a sus necesidades por medio del método de estandarización de datos. Con estos datos, se realizaron todos los procesos descritos en el diseño del sistema: selección de hiperparametros, muestreo, entrenamiento, hibridación y evaluación. Por último se emplea el modelo de priorización de algoritmos y se encuentra que una aproximación básica del algoritmo de filtrado colaborativo (surprise_BaselineOnly) es la que mejor lo hace para las métricas y los pesos seleccionados (RMSE, Recall y tiempo), para el conjunto de datos de Movielens.

Para este caso se encuentra que las hibridaciones no dieron mejores resultados que los dos algoritmos que conforman la hibridación. Por otro lado, la prueba de significancia estadística permitió identificar que existen diferencias estadísticas entre los primeros resultados de la priorización de algoritmos. Por lo que para la población, es muy probable que siempre queden los mismos algoritmos en los primeros lugares.

Amazon

Descripción: Amazon.com es una compañía estadounidense de comercio electrónico. Varios conjuntos de datos que representan interacciones entre las diferentes categorias de productos de Amazon.com han sido compartidas para la comunidad científica y en general. En esta prueba se usará un conjunto de datos que representa las interacciones de la categoria Dulces y comida gourmet.

Dominio: Productos.

Conjunto de datos:

- Nombre archivo: ratings_Grocery_and_Gourmet_Food.csv
- Número de interacciones: 1297156
- Peso del conjunto de datos: 52 MB
- Busca recomendar: Productos.
- Información incluida: userId,productId,rating,timestamp
- Formato del archivo: Los archivos del conjunto de datos están codificados en UTF-8, son archivos separados por comas y no contiene una fila con las cabeceras de los datos. Estos archivos se encuentran en formato CSV.
- Filas: 1297156
- Columnas: 4
- Tipo de rating: explícito – númerico – Entero
- Rango de rating: (1 – 5)
- Configuración del cliente

```
[EVALUATION]
precision_weight = 0
recall_weight = 0
time_weight = 0.1
rmse_weight = 0.45
mae_weight = 0.45

[RESULTS]
```

```
statistical_significance = true
number_of_2_fold_samples = 20
min_population_constraint = 50000
test_metric = total

[CLEANING]
min_interactions_explicit = 4

[SAMPLING]
number_of_folds = 1
test_size = 0.25
explicit_rating_threshold_for_test = 4
maximum_interactions_evaluation = 500000

[RECOMMEND]
recommendations_per_user=20

[HYBRIDATION]
hybridation_threshold = 0.8
max_for_hybrid = 5
```

Para este cliente se activa la prueba de significancia estadística (statistical_significance), lo que permitirá conocer la confianza en la priorización de los algoritmos. De acuerdo a lo anterior, en la etapa de sampling no se usará la técnica K-fold sino que se tomarán varias muestras de la población y sobre cada una de estas se tomará un 25% para pruebas y un 75% para entrenamiento, como indica la propiedad (test_size). La propiedad (explicit_rating_threshold_for_test) indica el valor de rating mínimo que se debe tomar para el conjunto de pruebas y la propiedad (maximum_interactions_evaluation) indica el tamaño máximo de cada muestra.

En Cleaning se indica que solo se tendrán en cuenta usuarios con al menos 4 interacciones para el entrenamiento de los algoritmos. Esta decisión se debe al conocimiento que los algoritmos de filtro colaborativo funcionan mejor para usuarios con al menos 3 interacciones.

Se generan listas de recomendaciones de 20 artículos para cada usuario en el conjunto de pruebas como indica la propiedad (recommendations_per_user). Estas listas permiten evaluar las métricas de uso de la recomendación y generar los listados de recomendaciones en producción.

Para la hibridación se tomarán los algoritmos con un valor total superior a 0.8 y se tomarán como máximo 5 algoritmos.

En la evaluación de los algoritmos se define un interés en las métricas MAE en un 45%, RMSE en un 45% y tiempo de entrenamiento en un 10%, indicando un interés total en métricas relacionadas con la exactitud en la predicción de la calificación, dando igual importancia a los aspectos penalizados por RMSE como a los de MAE. Por otro lado, el tiempo de entrenamiento se considera solo un poco importante.

Proceso ETL:

- Configuración para el mapeo (ETL): Este archivo contiene la configuración definida para el cliente y que permite representar el formato original de los datos del cliente hacia el formato usado por el sistema.

```
{
    "input_type":"explicit",
    "is_json": false,
    "file_type_information": {
        "csv_headers": false,
        "csv_separator":",",
    },
    "key_map": {
        "rating":2,
        "user_id":0,
        "item_id":1,
        "timestamp":3,
        "event_side_features":"",
        "rating_config": {
            "min_rating":1,
            "max_rating":5,
        },
```

```
        "remove_content":"",
    }
}
```

Este cliente aunque tiene sus datos en archivos con formato CSV, a diferencia del cliente anterior, no tiene cabeceras de los datos. Por lo tanto, se debe especificar al sistema la posición de cada columna para que este la represente a los datos usados por los algoritmos.

- Reglas de limpieza aplicadas: Se retiran los usuarios con menos de 4 interacciones para la selección de algoritmos de filtro colaborativo, ya que este tipo de algoritmos no responden bien para usuarios con pocas interacciones. Al entrenar el algoritmo seleccionado se deben tener en cuenta todos los usuarios y recomendar los artículos más populares a los usuarios con pocas interacciones.
- Formato resultante y características de los datos:

index	target	user_id	timestamp	item_id
0	5.0	A1ZQZ8RJS1XVTX	1381449600	0657745316
1	5.0	A31W38VGZAUUM4	1354752000	0700026444
2	1.0	A3I0AV0UJX5OH0	1385942400	1403796890
3	3.0	A3QAAOLIXKV383	1307836800	1403796890
4	5.0	AB1A5EGHHVA9M	1332547200	141278509X
5	1.0	A3DTB6RVENLQ9Q	1362268800	1453060375
6	3.0	A3LZA698SQPCXE	1374019200	1453060464
7	4.0	A2XZPK86YY9R6G	1376956800	1453060782
8	3.0	A2MWO0CISKXJ9	1391904000	1603112251
9	5.0	ACDUAY8AH3T72	1388534400	1613170416
10	4.0	A11WU9TYDLB1RG	1392163200	1613170416

Figura 14: Dataframe resultante - Amazon

Luego de ejecutar el proceso de ETL se obtiene un Dataframe de Pandas (figura 14). Este corresponde al formato general usado por los algoritmos de recomendación del sistema.

- Número de usuarios: 768438 usuarios.
- Número de artículo: 166049 artículos.
- Número de ratings: 1297156
- Porcentaje de valores vacíos en la matriz: 0.9999898340700841
- Descripción estadística del conjunto de datos:

	target
count	1.297156e+06
mean	4.254563e+00
std	1.253953e+00
min	1.000000e+00
25%	4.000000e+00
50%	5.000000e+00
75%	5.000000e+00
max	5.000000e+00

Tabla 3: Descripción estadística - Amazon

- Distribución de las calificaciones en el conjunto de datos:

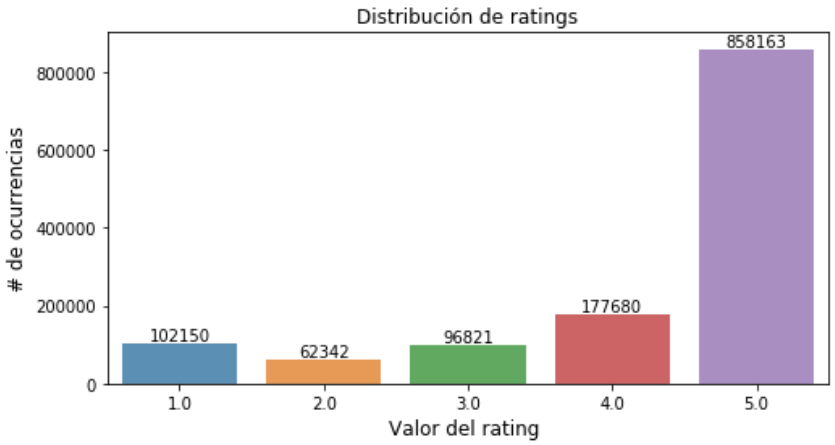

Figura 15: Distribución de ratings – Amazon

La información anterior nos indica que hay muchos más usuarios que artículos. Por lo tanto es más óptimo computacionalmente para las técnicas de filtro colaborativo basadas en memoria, usar un enfoque orientado en el artículo, ya que un enfoque basado en usuario va a requerir cargar una matriz más grande en memoria. La

matriz de interacciones está conformada en un 99% por valores vacíos, lo cual es común en este tipo de conjuntos de datos. Aun así, de acuerdo a la densidad de la matriz algunos algoritmos podrían presentar mejores resultados.

En la tabla 3 se presenta la descripción estadística del conjunto de datos, incluyendo valores como la media y la desviación estándar. En la figura 15 se muestra la distribución de los ratings. Según estos datos se puede ver que la mayoría de las calificaciones toman valores entre 4 y 5, la media es 4.25 y la desviación es de 1.25. esto indica que los usuarios del conjunto tienden a dar calificaciones muy altas a los artículos y por lo tanto hay más información que le permita a los algoritmos aprender sobre lo que le gusta a los usuarios pero no hay tanta información que represente lo que no les gusta, lo cual puede sesgar al modelo dificultando el distinguir lo que no le gusta al usuario.

Proceso de entrenamiento y evaluación de algoritmos:

El conjunto de datos obtenido en el paso anterior es usado para entrenar y evaluar los algoritmos del sistema. Los resultados (sin normalizar) para cada algoritmo en las diferentes métricas son los siguientes:

model	mae	precision	recall	rmse	time
surprise_svd	0.80145158	0.86322077	0.86850347	1.06910564	15.3656387
surprise_BaselineOnly	0.81436228	0.85240135	0.90524371	1.06960766	1.34473745
surprise_SVDpp	0.79136	0.87210394	0.84577951	1.06747281	76.2470553
surprise_CoClustering	0.8110288	0.87529563	0.79042666	1.17834696	13.8816083
surprise_KNNBaseline	0.81166708	0.85596424	0.88206616	1.11000049	153.160389
surprise_SlopeOne	0.82696173	0.86596943	0.81490404	1.18220038	160.144266
surprise_KNNBasic	0.86658821	0.8279401	0.94101234	1.16781687	152.566081
surprise_KNNWithMeans	0.84084875	0.85160711	0.85055091	1.20656949	153.513724
surprise_KNNWithZScore	0.83910217	0.85055688	0.85433095	1.20892896	155.782996
surprise_NMF	0.9901497	0.89124955	0.63790583	1.2837552	23.558608
surprise_NormalPredictor	1.08130777	0.8518154	0.63554093	1.45887283	0.28940941

Teniendo en cuenta los pesos definidos en la configuración del cliente y luego de normalizar los valores de cada métrica, se aplica el modelo de priorización y se obtienen los siguientes resultados totales para los algoritmos independientes:

model	norm_mae	norm_rmse	norm_time	total
surprise_svd	0.96519516	0.99582824	0.90568801	0.97302933
surprise_BaselineOnly	0.92066749	0.9945456	0.99339821	0.96118571
surprise_SVDpp	1	1	0.52483367	0.95248337
surprise_CoClustering	0.93216432	0.7167242	0.91497162	0.833497
surprise_KNNBaseline	0.92996298	0.89134472	0.04368887	0.82395735
surprise_SlopeOne	0.87721331	0.70687899	0	0.71284153
surprise_KNNBasic	0.74054564	0.74362785	0.04740666	0.67261874
surprise_KNNWithMeans	0.82931839	0.64461761	0.04147852	0.66741905
surprise_KNNWithZScore	0.83534218	0.63858931	0.02728269	0.66599744
surprise_NMF	0.31439479	0.44741345	0.85443546	0.42825726
surprise_NormalPredictor	0	0	1	0.1

Cada métrica debe aportar en un porcentaje al valor total que permite priorizar los algoritmos. Algunos valores como precisión y recall son mejores mientras estén más cerca de 1. Por otro lado, valores como MAE, RMSE y time son mejores mientras estén más cerca de 0. Es por esto que MAE, RMSE y time son normalizados a valores entre 0 y 1, y su valor original es invertido para que el aporte al total sea correspondiente a su cercanía a 0.

Proceso de hibridación de algoritmos:

Según la configuración del cliente, para la hibridación se tomarán como máximo 5 algoritmos que tengan un valor total por encima de 0.8, suponiendo que para el proveedor son relevantes para hibridación todos los algoritmos que lo hagan al menos un 80% bien en las métricas de evaluación. De acuerdo con esto se seleccionan los siguientes algoritmos:

['surprise_svd', 'surprise_BaselineOnly', 'surprise_SVDpp', 'surprise_CoClustering', 'surprise_KNNBaseline']

La hibridación y evaluación de los algoritmos seleccionados presenta los siguientes resultados (sin normalizar):

model 1	model 2	mae	precision	recall	rmse	time
surprise_svd	surprise_SVDpp	0.7948427	0.86839205	0.86067042	1.06413007	91.6126941
surprise_SVDpp	surprise_KNNBaseline	0.79545875	0.86731808	0.86131182	1.06930666	229.407444
surprise_BaselineOnly	surprise_SVDpp	0.80138855	0.86398977	0.87469149	1.06427599	77.5917928
surprise_SVDpp	surprise_CoClustering	0.79151115	0.87872787	0.81152128	1.08801325	90.1286636
surprise_svd	surprise_CoClustering	0.79587724	0.87459528	0.82001216	1.08404493	29.247247
surprise_svd	surprise_KNNBaseline	0.80103622	0.86178939	0.87473574	1.07059304	168.526027
surprise_svd	surprise_BaselineOnly	0.80699694	0.85839199	0.88846047	1.06699874	16.7103762
surprise_BaselineOnly	surprise_CoClustering	0.80254187	0.87034464	0.82839529	1.08227915	15.2263457
surprise_BaselineOnly	surprise_KNNBaseline	0.80847473	0.8560022	0.89074554	1.07190167	154.505126
surprise_CoClustering	surprise_KNNBaseline	0.79914844	0.87155245	0.81971934	1.09927602	167.041997

Teniendo en cuenta los pesos definidos en la configuración del cliente y luego de normalizar los valores de cada métrica, se aplica el modelo de priorización y se obtienen los siguientes resultados totales para las hibridaciones:

model 1	model 2	norm_mae	norm_rmse	norm_time	total
surprise_svd	surprise_SVDpp	0.80360562	1	0.64335626	0.87595815
surprise_SVDpp	surprise_KNNBaseline	0.76728976	0.85271146	0	0.72900055
surprise_BaselineOnly	surprise_SVDpp	0.41772901	0.99584812	0.70881909	0.70699162
surprise_SVDpp	surprise_CoClustering	1	0.32045725	0.65028512	0.65923427
surprise_svd	surprise_CoClustering	0.74261962	0.43336684	0.93453717	0.62264762
surprise_svd	surprise_KNNBaseline	0.43849886	0.81611061	0.28425205	0.59299946
surprise_svd	surprise_BaselineOnly	0.08711505	0.9183783	0.99307114	0.55177912
surprise_BaselineOnly	surprise_CoClustering	0.34974098	0.48360825	1	0.47500715
surprise_BaselineOnly	surprise_KNNBaseline	0	0.77887639	0.34971488	0.38546586
surprise_CoClustering	surprise_KNNBaseline	0.54978323	0	0.29118091	0.27652055

Priorización y selección de algoritmos:

model	norm_mae	norm_rmse	norm_time	total
surprise_svd	0.96519516	0.98739541	0.93419885	0.972085642
surprise_svd_surprise_BaselineOnly	0.94606979	0.99273281	0.92832966	0.965294135
surprise_SVDpp	1	0.99153184	0.6684781	0.96303714
surprise_svd_surprise_CoClustering	0.98442048	0.94954977	0.87361171	0.957647785
surprise_BaselineOnly	0.92066749	0.98612364	0.99539395	0.957595404
surprise_BaselineOnly_surprise_CoClustering	0.96143488	0.95402302	0.93480681	0.955436735
surprise_svd_surprise_SVDpp	0.98798851	1	0.60141381	0.954736209
surprise_BaselineOnly_surprise_SVDpp	0.96541256	0.99963034	0.66260891	0.950530195
surprise_SVDpp_surprise_CoClustering	0.99947868	0.93949686	0.60789095	0.933328087
surprise_svd_surprise_KNNBaseline	0.96662771	0.98362739	0.26572075	0.904186873
surprise_BaselineOnly_surprise_KNNBaseline	0.94097307	0.98031224	0.32691585	0.897269976
surprise_SVDpp_surprise_KNNBaseline	0.98586383	0.98688615	0	0.88773749
surprise_CoClustering_surprise_KNNBaseline	0.97313847	0.91096492	0.2721979	0.875066317
surprise_KNNBaseline	0.92996298	0.88379668	0.33278504	0.849470351
surprise_CoClustering	0.93216432	0.71065487	0.940676	0.833336235
surprise_SlopeOne	0.87721331	0.70089303	0.30230347	0.740378199
surprise_KNNBasic	0.74054564	0.7373307	0.33537894	0.698582244
surprise_KNNWithMeans	0.82931839	0.63915888	0.33124289	0.69393906
surprise_KNNWithZScore	0.83534218	0.63318164	0.32133851	0.69296957

surprise_NMF	0.31439479	0.44362468	0.89844013	0.430952778
surprise_NormalPredictor	0	0	1	0.1

La normalización de las métricas depende de los valores de los otros algoritmos con los que se compara para la priorización. Es por esto, que al agregar las hibridaciones a los resultados y aplicar de nuevo el modelo de priorización, los valores totales cambian con respecto a los totales de los algoritmos individuales o hibridaciones por separado.

Prueba estadística t-test:

El sistema realiza la prueba estadística sobre una métrica definida por el usuario, para este cliente se usa la métrica: total, lo que indica que sobre todas las muestras tomadas por cada algoritmo, se usará el valor de total para la prueba t-test.

Se realiza la prueba estadística sobre las tres primeras posiciones de los resultados para conocer la probabilidad de que la primera posición sea reemplazada por alguna de las dos siguientes. A continuación se presenta el resultado:

Muestras para total:

Posición 1: surprise_svd

[0.96294483 0.9657318 0.96457108 0.96076224 0.96528893 0.96075782
 0.95797044 0.96460163 0.97181806 0.95767191 0.96662168 0.9633044
 0.97228821 0.95907899 0.97072901 0.96447021 0.9538448 0.96815065
 0.96231107 0.96217025]

Posición 2: surprise_svd_surprise_BaselineOnly

 [0.95557663 0.95976078 0.95783905 0.95371459 0.95873195 0.95408184
 0.95133247 0.95790569 0.96459397 0.9513407 0.96070348 0.95699299
 0.96561125 0.95258526 0.96404926 0.95851664 0.94822618 0.96110338
 0.955213   0.95630279]

Posición 3: surprise_SVDpp

[0.95316092 0.95814275 0.9556788 0.95065675 0.95547507 0.95185286
 0.95074038 0.95781333 0.96173705 0.95001297 0.96024658 0.95114531
 0.95984704 0.94799821 0.96102945 0.95317496 0.94303191 0.9566906
 0.95139411 0.95202259]

Resultado:

Posición 1 – Posición 2:

statistic=4.383138339647608

pvalue=8.920681821082499e-05

Posición 2 – Posición 3:

statistic=-2.075993849397193

pvalue=0.04471075685820751

El valor de p-value para ambos casos es menor que 0.05 por lo tanto se rechaza la hipótesis nula y se concluye que existe significancia estadística en la diferencia de las muestras. Se asume que para la población se mantendrán los mismos algoritmos en la posición 1 y la posición 2.

En este caso, se evaluó el prototipo con un conjunto de datos relacionado con el dominio de productos, específicamente alimentos de supermercado. Para este conjunto de datos, los datos de entrada se encontraban en archivos con formato CSV separado por comas y sin cabeceras. En este caso era necesario identificar la posición de los datos originales por medio de valores numéricos en la configuración de la fuente de datos.

El sistema transformó correctamente los datos de entrada a sus necesidades por medio del método de estandarización de datos. Con estos datos, se realizaron todos los procesos descritos en el diseño del sistema: selección de hiperparametros, muestreo, entrenamiento, hibridación y evaluación. Por último se emplea el modelo de priorización de algoritmos y se encuentra que un algoritmo de factorización matricial (surprise_svd) es el que mejor lo hace para las métricas y los pesos seleccionados (RMSE, MAE y tiempo), para el conjunto de datos de Amazon (Grocery_and_Gourmet_Food).

Para este caso se encuentra que las hibridaciones no dieron mejores resultados que los dos algoritmos que conforman la hibridación. Por otro lado, la prueba de significancia estadística permitió identificar que existen diferencias estadísticas entre los primeros resultados de la priorización de algoritmos. Sin embargo, el valor *p* para el segundo y tercer algoritmo no es tan pequeño como en el caso de Movielens. Por lo tanto, solo hay una confianza del 95% de que para la población, el segundo y tercer algoritmo mantengan la posición correspondiente.

Yelp

Descripción: Yelp es un servicio para búsqueda de negocios que permite a los usuarios, según sus intereses, encontrar, reservar y calificar negocios en diferentes aspectos. Yelp pone a disposición de la comunidad un subconjunto de los datos de su negocio incluyendo calificaciones y comentarios que dejan los usuarios a los negocios.

Dominio: Negocios - lugares.

Conjunto de datos:

- Nombre archivo: yelp_review.json
- Número de interacciones: 5261668
- Peso del conjunto de datos: 3.7 GB
- Busca recomendar: Negocios.
- Información incluida(columnas): review_id, user_id, business_id, stars, date, text, useful, funny, cool.
- Formato del archivo: Los archivos del conjunto de datos se encuentran en formato JSON y cada línea corresponde a un registro.
- Filas: 5261668
- Columnas: 9
- Tipo de rating: explícito – numérico – Entero
- Rango de rating: (0 – 5)
- Configuración del cliente

```
[EVALUATION]
precision_weight = 0.2
recall_weight = 0.2
time_weight = 0.2
rmse_weight = 0.2
mae_weight = 0.2

[RESULTS]
statistical_significance = false
number_of_2_fold_samples = 20
```

```
min_population_constraint = 50000
test_metric = total


[CLEANING]
min_interactions_explicit = 4

[SAMPLING]
number_of_folds = 5
test_size = 0.25
explicit_rating_threshold_for_test = 4
maximum_interactions_evaluation = 500000

[RECOMMEND]
recommendations_per_user=20

[HYBRIDATION]
hybridation_threshold = 0.8
max_for_hybrid = 4
```

Para este cliente se define usar los valores por defecto del sistema. Por lo tanto, no se activa la significancia estadística (statistical_significance), esto permitirá aplicar el escenario de muestreo donde se usa la técnica k-fold para la obtención de los conjuntos de entrenamiento y pruebas, usando como *k* el valor definido en la propiedad number_of_folds.

En Cleaning se indica que solo se tendrán en cuenta usuarios con al menos 4 interacciones para el entrenamiento de los algoritmos. Esta decisión se debe al conocimiento que los algoritmos de filtro colaborativo funcionan mejor para usuarios con al menos 3 interacciones.

Se generan listas de recomendaciones de 20 artículos para cada usuario en el conjunto de pruebas como indica la propiedad (recommendations_per_user). Estas listas permiten evaluar las métricas de uso de la recomendación y generar los listados de recomendaciones en producción.

Para la hibridación se tomarán los algoritmos con un valor total superior a 0.8 y se tomarán como máximo 4 algoritmos (valor por defecto).

En la evaluación de los algoritmos se define un interés en todas las métricas por igual (valor por defecto). Esto ocurre cuando el proveedor o cliente no tiene una preferencia por una o varias métricas. Por el contrario, se quiere priorizar los algoritmos, de acuerdo a que tan bien lo hacen en general para todas las métricas evaluadas.

Proceso ETL:

- Configuración para el mapeo (ETL): Este archivo contiene la configuración definida para el proveedor y que permite representar el formato original de los datos del cliente hacia el formato usado por el sistema.

```
{
    "input_type":"explicit",
    "is_json": true,
    "file_type_information": {
        "json_lines": true,
        "json_no_lines_record_path":"",
        "json_no_lines_meta": ["",""]
    },
    "key_map": {
        "rating":"stars",
        "user_id":"user_id",
        "item_id":"business_id",
        "timestamp":"date",
        "event_side_features": {
            "useful":"useful",
            "funny":"funny",
            "cool":"cool"
        },
        "rating_config": {
            "min_rating":1,
            "max_rating":5,
```

```
        },
        "remove_content":"",
    }
```

Para este cliente se decide incluir otra información además de la básica usada por los algoritmos. Esta información se considera como características que acompañan el evento (event_side_features) y el sistema propuesto permite agregar toda la información que se considere necesaria. Sin embargo, el prototipo no hará uso de esta información ya que los algoritmos incluidos no están diseñados para consumir estos datos adicionales. Para trabajos futuros se propone incluir algoritmos que usen esta información y así darle un uso apropiado.

Para este cliente se define la propiedad is_json:true, lo que le indica al sistema que los archivos estan en formato JSON. La propiedad json_lines indica que los registros en el archivo JSON se encuentran separados por líneas.

- Reglas de limpieza aplicadas: Se retiran los usuarios con menos de 4 interacciones para la selección de algoritmos de filtro colaborativo, ya que este tipo de algoritmos no responden bien para usuarios con pocas interacciones. Al entrenar el algoritmo seleccionado se deben tener en cuenta todos los usuarios y recomendar los artículos más populares a los usuarios con pocas interacciones.
- Formato resultante y características de los datos:

index	target	user_id	funny	useful	cool	timestamp	item_id
0	5	bv2nCi5Qv5vroFiqKGopiw	0	0	0	2016-05-28	AEx2SYEUJmTxVVB18LlCwA
1	5	bv2nCi5Qv5vroFiqKGopiw	0	0	0	2016-05-28	VR6GpWIda3SfvPC-lg9H3w
2	5	bv2nCi5Qv5vroFiqKGopiw	0	0	0	2016-05-28	CKC0-MOWMqoeWf6s-szl8g
3	4	bv2nCi5Qv5vroFiqKGopiw	0	0	0	2016-05-28	ACFtxLv8pGrrxMm6EgjreA
4	4	bv2nCi5Qv5vroFiqKGopiw	0	0	0	2016-05-28	s2I_Ni76bjJNK9yG60iD-Q
5	5	_4iMDXbXZ1p1ONG297YEAQ	0	1	0	2014-09-24	8QWPlVQ6D-OExqXoaD2Z1g
6	4	u0LXt3Uea_GidxRW1xcsfg	0	0	2	2012-05-11	9_CGhHMz8698M9-PkVf0CQ
7	4	u0LXt3Uea_GidxRW1xcsfg	0	1	0	2015-10-27	gkCorLgPyQLsptTHalL61g
8	3	u0LXt3Uea_GidxRW1xcsfg	0	1	0	2013-02-09	5r6-G9C4YLbC7Ziz57l3rQ
9	5	u0LXt3Uea_GidxRW1xcsfg	0	3	0	2016-04-06	fDF_o2JPU8BR1Gya--jRIA
10	4	u0LXt3Uea_GidxRW1xcsfg	0	1	0	2013-05-01	z8oIoCT1cXz7gZP5GeU5OA
11	3	u0LXt3Uea_GidxRW1xcsfg	0	5	1	2011-09-28	XWTPNfskXoUL-Lf32wSk0Q
12	1	u0LXt3Uea_GidxRW1xcsfg	0	9	1	2011-02-16	13nKUHH-uEUXVZylgxchPA
13	3	u0LXt3Uea_GidxRW1xcsfg	1	2	1	2012-12-03	RtUvSWO_UZ8V3Wpj0n077w
14	5	u0LXt3Uea_GidxRW1xcsfg	0	2	1	2010-07-16	Aov96CM4FZAXeZvKtsStdA
15	4	u0LXt3Uea_GidxRW1xcsfg	0	0	0	2011-09-28	0W4lkclzZThpx3V65bVgig
16	1	u0LXt3Uea_GidxRW1xcsfg	0	0	0	2012-10-23	fdnNZMk1NP7ZhL-YMidMpw
17	3	u0LXt3Uea_GidxRW1xcsfg	0	2	0	2010-09-15	PFPUMF38-lraKzLcTiz5gQ
18	3	u0LXt3Uea_GidxRW1xcsfg	0	4	0	2012-09-23	oWTn2IzrprsRkPfULtjZtQ
19	1	u0LXt3Uea_GidxRW1xcsfg	2	9	1	2012-10-30	zgQHtqX0gqMw1nlBZl2VnQ

Figura 16: Dataframe resultante - Yelp

Luego de ejecutar el proceso de ETL se obtiene un Dataframe de Pandas (figura 16). Este corresponde al formato general usado por los algoritmos de recomendación del sistema.

- Número de usuarios: 1326101 usuarios.
- Número de artículos: 174567 artículos.
- Número de ratings: 5261668
- Porcentaje de valores vacíos en la matriz: 0.9999772707717166
- Descripción estadística del conjunto de datos:

	target
count	5.261668e+06
mean	3.727739e+00
std	1.433593e+00
min	1.000000e+00
25%	3.000000e+00
50%	4.000000e+00
75%	5.000000e+00
max	5.000000e+00

Tabla 4: Descripción estadística - Yelp

- Distribución de las calificaciones en el conjunto de datos:

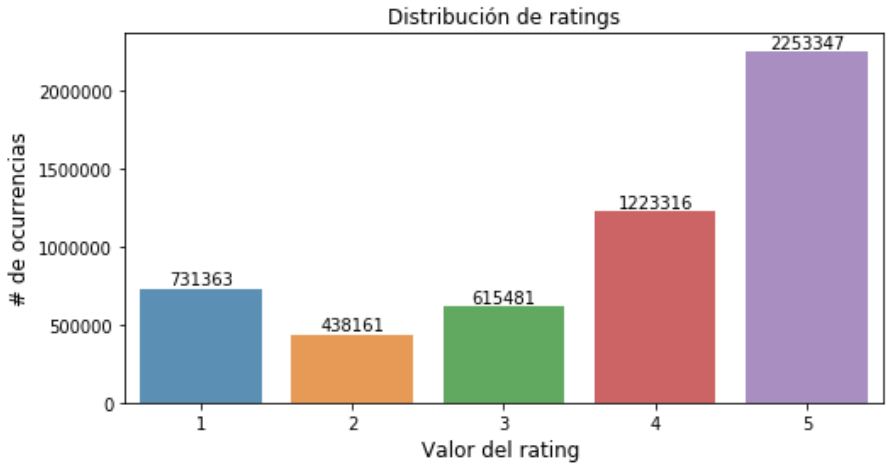

Figura 17: Distribución de ratings - Yelp

Al igual que en los casos anteriores, la información anterior nos indica que hay muchos más usuarios que artículos. Por lo tanto es más óptimo computacionalmente para las técnicas de filtrado colaborativo basadas en memoria, usar un enfoque orientado en el artículo, ya que un enfoque basado en usuario va a requerir cargar una matriz más grande en memoria. La matriz de interacciones está conformada en un 99% por valores vacíos, lo cual es común en este tipo de conjuntos de datos. Aun así, de acuerdo a la densidad de la matriz algunos algoritmos podrían presentar mejores resultados.

En la tabla 3 se presenta la descripción estadística del conjunto de datos, incluyendo valores como la media y la desviación estándar. En la figura 15 se muestra la distribución de los ratings. Según estos datos se puede ver que la mayoría de las calificaciones toman valores entre 4 y 5, la media es 4.25 y la desviación es de 1.25. esto indica que los usuarios del conjunto tienden a dar calificaciones muy altas a los artículos y por lo tanto hay más información que le permita a los algoritmos aprender sobre lo que le gusta a los usuarios pero no hay tanta información que

represente lo que no les gusta, lo cual puede sesgar al modelo dificultando el distinguir lo que no le gusta al usuario.

En la tabla 4 se presenta la descripción estadística del conjunto de datos, incluyendo valores como la media y la desviación estándar. En la figura 17 se muestra la distribución de los ratings. Según estos datos se puede ver que la mayoría de los ratings toman valores entre 4 y 5, la media es 3.73 y la desviación es de 1.43. esto indica que los usuarios del conjunto tienden a dar calificaciones altas a los artículos y por lo tanto hay más información que le permita a los algoritmos aprender sobre lo que le gusta a los usuarios pero no hay tanta información que represente lo que no les gusta, lo cual puede sesgar al modelo dificultando el distinguir lo que no le gusta al usuario. Sin embargo, este cliente tiene más distribuidas las calificaciones que los dos clientes anteriores.

Proceso de entrenamiento y evaluación de algoritmos:

El conjunto de datos obtenido en el paso anterior es usado para entrenar y evaluar los algoritmos del sistema. Los resultados (sin normalizar) para cada algoritmo en las diferentes métricas son los siguientes:

model	mae	precision	recall	rmse	time
surprise_svd	1.02359146	0.95378368	0.47839459	1.25803127	2.17678881
surprise_BaselineOnly	1.02038207	0.94205051	0.51126338	1.25848365	21.7030551
surprise_SVDpp	1.01457441	0.93572498	0.53456657	1.25443764	80.3301864
surprise_KNNBaseline	1.04810673	0.9439559	0.48977955	1.29528985	238.016712
surprise_CoClustering	1.07701175	0.91439131	0.55142401	1.41486085	23.7033267
surprise_KNNWithMeans	1.06679293	0.9101652	0.54349121	1.36511563	241.239265
surprise_KNNWithZScore	1.06862119	0.90944725	0.54485322	1.36775755	244.584182
surprise_KNNBasic	1.12369218	0.96855025	0.3258107	1.37318748	238.95172
surprise_NMF	1.17843816	0.95749877	0.41790178	1.47205947	38.6299095
surprise_SlopeOne	1.13844314	0.89268078	0.55560247	1.48528329	252.418854
surprise_NormalPredictor	1.38162607	0.82941377	0.57779784	1.74146993	0.42623186

Teniendo en cuenta los pesos definidos en la configuración del cliente y luego de normalizar los valores de cada métrica, se aplica el modelo de priorización y se obtienen los siguientes resultados totales para los algoritmos independientes:

model	norm_mae	norm_rmse	norm_time	total
surprise_svd	0.97543385	0.99262138	0.99305314	0.87865733
surprise_BaselineOnly	0.98417753	0.99169254	0.91556569	0.86894993
surprise_SVDpp	1	1	0.68291153	0.83064062
surprise_CoClustering	0.82989495	0.67061073	0.90762787	0.77478977
surprise_NMF	0.5535676	0.55316756	0.84839367	0.66610588
surprise_KNNBaseline	0.90864414	0.91612014	0.05715303	0.66313055
surprise_KNNWithMeans	0.85773524	0.7727502	0.04436475	0.62570132
surprise_KNNWithZScore	0.8527543	0.76732568	0.03109088	0.62109427
surprise_KNNBasic	0.70271821	0.75617667	0.05344257	0.56133968
surprise_SlopeOne	0.66253054	0.52601573	0	0.5273659
surprise_NormalPredictor	0	0	1	0.48144232

Cada métrica debe aportar en un porcentaje al valor total que permite priorizar los algoritmos. Algunos valores como precisión y recall son mejores mientras están más cerca de 1. Por otro lado, valores como MAE, RMSE y time son mejores mientras estén más cerca de 0. Es por esto que MAE, RMSE y time son normalizados a valores entre 0 y 1, y su valor original es invertido para que el aporte al total sea correspondiente a su cercanía a 0.

Proceso de hibridación de algoritmos:

Según la configuración del cliente, para la hibridación se tomarán como máximo 4 algoritmos que tengan un valor total por encima de 0.8, suponiendo que para el proveedor son relevantes para hibridación todos los algoritmos que lo hagan al menos un 60% bien en las métricas de evaluación. De acuerdo con esto se seleccionan los siguientes algoritmos:

['surprise_svd', 'surprise_BaselineOnly', 'surprise_SVDpp']

La hibridación y evaluación de los algoritmos seleccionados presenta los siguientes resultados:

model 1	model 2	mae	precision	recall	rmse	time
surprise_svd	surprise_SVDpp	1.01531448	0.94080268	0.52080548	1.25363615	102.033242
surprise_BaselineOnly	surprise_SVDpp	1.01717238	0.94536766	0.50735347	1.2533764	82.5069752
surprise_BaselineOnly	surprise_svd	1.02080943	0.94862785	0.49489532	1.25664611	23.879844

Teniendo en cuenta los pesos definidos en la configuración del cliente, se obtienen los siguientes resultados totales para las hibridaciones:

model 1	model 2	norm_mae	norm_rmse	norm_time	total
surprise_svd	surprise_SVDpp	1	0.92056019	0	0.67643367
surprise_BaselineOnly	surprise_SVDpp	0.6618894	1	0.24984539	0.67289119
surprise_BaselineOnly	surprise_svd	0	0	1	0.48870463

Selección de algoritmo:

model	norm_mae	norm_rmse	norm_time	total
surprise_svd	0.97543385	0.99046316	0.99305314	0.87822569
surprise_BaselineOnly	0.98417753	0.98953634	0.91556569	0.86851869
surprise_BaselineOnly_surprise_svd	0.98301324	0.99330105	0.90692739	0.86535297
surprise_SVDpp	1	0.99782574	0.68291153	0.83020576
surprise_BaselineOnly_surprise_SVDpp	0.99292206	1	0.67427323	0.82398328
surprise_svd_surprise_SVDpp	0.99798375	0.99946784	0.59678578	0.8111691
surprise_CoClustering	0.82989495	0.66915265	0.90762787	0.77449816
surprise_NMF	0.5535676	0.55196483	0.84839367	0.66586533
surprise_KNNBaseline	0.90864414	0.91412826	0.05715303	0.66273218
surprise_KNNWithMeans	0.85773524	0.77107004	0.04436475	0.62536529
surprise_KNNWithZScore	0.8527543	0.76565731	0.03109088	0.62076059
surprise_KNNBasic	0.70271821	0.75453254	0.05344257	0.56101086
surprise_SlopeOne	0.66253054	0.52487204	0	0.52713716
surprise_NormalPredictor	0	0	1	0.48144232

La normalización de las métricas depende de los valores de los otros algoritmos con los que se compara para la priorización. Es por esto, que al agregar las

hibridaciones a los resultados y aplicar de nuevo el modelo de priorización, los valores totales cambian con respecto a los totales de los algoritmos individuales o hibridaciones por separado.

En este caso, se evaluó el prototipo con un conjunto de datos relacionado con el dominio de negocios y en donde los datos de entrada se encontraban en archivos con formato JSON separado por líneas. Se dejaron los parámetros de configuración del sistema en sus valores por defecto, incluyendo un peso igual para todas las métricas y no se usa la prueba de significancia estadística por los conjuntos de entrenamiento y pruebas se consiguen por medio de la técnica k-fold.

El sistema transformó correctamente los datos de entrada a sus necesidades por medio del método de estandarización de datos. Con estos datos, se realizaron todos los procesos descritos en el diseño del sistema: selección de hiperparametros, muestreo, entrenamiento, hibridación y evaluación. Por último se emplea el modelo de priorización de algoritmos y se encuentra que un algoritmo de factorización matricial (surprise_svd) es el que mejor lo hace para las métricas y los pesos seleccionados (Todas las métricas), para el conjunto de datos de Yelp.

Para este caso se encuentra que los algoritmos a hibridar son menos de los permitidos por la configuración del sistema, esto se debe a que el cuarto algoritmo en la lista de priorización no cumple con el umbral definido de 0.8 en el valor total. Las hibridaciones no dieron mejores resultados que los dos algoritmos que conforman la hibridación.

Luego de analizar los resultados totales en los algoritmos priorizados, se determina que sería útil emplear la prueba de significancia estadística en este conjunto de datos, ya que la diferencia en los valores totales para los primeros algoritmos de la lista no es significativa a simple vista.

ANÁLISIS DE RESULTADOS

Se emplean tres conjuntos de datos, correspondientes a los dominios: películas, productos y negocios. Estos conjuntos de datos comparten características como el porcentaje de datos vacíos en la matriz de ratings y el hecho de que los usuarios tienden a dar más calificaciones altas (positivas) a los artículos (distribución de calificaciones). Aparte de eso, cada conjunto de datos está descrito por medidas estadísticas diferentes, tienen diferentes cantidades de usuarios, artículos y calificaciones.

El formato de los archivos usados, es diferente para cada proveedor: CSV con encabezados, CSV sin encabezados, JSON separado por líneas. Por otro lado, cada conjunto de datos tiene una estructura, organización y contenido diferente. Con lo anterior, se consigue una diversidad de escenarios para validar que el prototipo del sistema propuesto pueda procesar diferentes conjuntos de datos.

Cada conjunto de datos es usado con el prototipo, iniciando con el método de estandarización de los datos, el cual depende de la configuración de la fuente de datos que es proporcionada por cada proveedor y que representa la estructura de los datos originales a la estructura de los datos requerida por el sistema para el uso de los algoritmos. El conjunto de datos estandarizado se almacena en la base de datos para el proveedor.

La selección de hiperparametros es un proceso que se ejecuta en paralelo, aun así, requiere un alto uso computacional y toma un tiempo considerable dependiendo del conjunto de datos y el conjunto de parámetros proporcionado para selección.

El entrenamiento de los algoritmos es en general rápido (menos de 5 minutos para el algoritmo más lento), aún cuando no se ejecuta en paralelo. Los algoritmos basados en memoria consumen muchos recursos según la cantidad de artículos o usuarios, dependiendo de si es basado en usuario o en artículo. Desde el estado del arte se reconoce la técnica basada en artículo como la más eficiente

computacionalmente y es capaz de producir recomendaciones más exactas para grandes conjuntos de artículos [2]. Por este motivo, y por la cantidad de usuarios vs. artículos, se usa la técnica basada en el artículo para todas las pruebas del prototipo.

La evaluación de los algoritmos requiere de varios procesos para el cálculo de cada métrica, especialmente de transformación y almacenamiento de datos. Al no estar optimizado para usar técnicas de procesamiento en paralelo, el sistema toma bastante tiempo realizando la evaluación de cada algoritmo. Es necesaria la optimización de este aspecto en trabajos futuros.

El algoritmo de factorización matricial ***SVD*** demostró tener el mejor resultado en dos de los escenarios de prueba. En algunas investigaciones del estado del arte [47][48], este y otros algoritmos basados en el modelo han mostrado tener un rendimiento superior sobre algoritmos basados en memoria para filtro colaborativo. El algoritmo ***Baseline only*** también fue seleccionado como el mejor en uno de los clientes y obtuvo el segundo puesto en los demás escenarios. La selección de algoritmos se debe al rendimiento de estos en las diferentes métricas y en la importancia que asigne el sistema según la configuración del proveedor.

La hibridación de algoritmos ocurre en relación a la configuración del cliente. En ninguno de los tres casos se presenta que un algoritmo híbrido sea seleccionado como el mejor, pero si es común que un algoritmo híbrido supere varios algoritmos simples e incluso otros algoritmos híbridos. La inclusión de otras fuentes de información como implícito o características de artículos y usuarios, y el uso de algoritmos en una técnica diferente, como basado en contenido, podría permitir una mejora superior sobre los resultados de hibridación.

El algoritmo seleccionado queda almacenado en la base de datos del proveedor. Una vez en producción, este algoritmo debe entrenarse con el conjunto de datos completo para generar un modelo que comience a atender las diferentes peticiones de recomendación. Este proceso al igual que la capa de servicios no fueron

implementados en el prototipo pero se describen en el diseño de la solución como los pasos a seguir.

CONCLUSIONES Y TRABAJOS FUTUROS

Los sistemas de recomendación son una herramienta cada vez más usada por diferentes plataformas y servicios que quieren proveer a sus clientes información sobre lo que podrían adquirir según sus gustos y comportamientos.

Un sistema de recomendación debe seguir varios procesos antes de llegar a generar recomendaciones para los clientes. La selección de hiperparámetros y entrenamiento de un algoritmo de recomendación son las tareas principales para conseguir una aproximación básica y generar recomendaciones.

La hibridación de algoritmos es ampliamente usada en el estado del arte para unir las fortalezas de dos o más algoritmos y mejorar las dificultades que pueden presentar independientemente. La mejoría en los resultados de los algoritmos hibridados ocurre principalmente al integrar diferentes técnicas o fuentes de información. En este trabajo no se encontró mejora en las hibridaciones realizadas por el uso de una misma técnica de recomendación y una sola fuente de información. Sin embargo, el proceso para hibridar algoritmos mantiene su importancia desde el diseño del sistema, para una futura aplicación de otras técnicas o fuentes de información.

Un sistema de recomendación que pueda ser usado por diferentes clientes con conjuntos de datos específicos de acuerdo al negocio, requiere un diseño que tenga en cuenta aspectos como la captura y transformación de los datos que alimentarán el sistema, ya que este sería un proceso dispendioso de aplicar manualmente a cada cliente nuevo. Otro aspecto importante se refiere a la selección de un algoritmo óptimo para cada conjunto de datos específico. El estado del arte ha demostrado que el rendimiento de un algoritmo de recomendación está determinado por las características del conjunto de datos específico.

CONCLUSIONES

Este trabajo diseña y válida mediante un prototipo un sistema de recomendación que puede proveer su servicio a diferentes clientes en diferentes dominios. Para esto, se consideran los siguientes aportes como los principales de este trabajo:

- Método de estandarización de datos: este método permite la obtención de los datos, su transformación y consistencia para ser consumidos por los algoritmos del sistema. diferentes negocios mantienen sus datos en diferentes formatos. Se diseña y prototipa un sistema ETL (extract-transform-load) que permite almacenar los archivos del cliente y extraerlos mediante una configuración proporcionada por éste, transformarlos y almacenarlos en un formato general que es usado por el resto del sistema. Se usa este método con tres casos de estudio diferentes y se obtienen los resultados esperados al permitir que los datos de cada cliente fueran transformados y usados por los algoritmos del sistema.
- Proceso de entrenamiento, hibridación y evaluación de algoritmos de recomendación. Para generar recomendaciones es necesario definir un algoritmo o varios para el caso de hibridación, entrenarlos y evaluarlos para buscar mejores resultados. Para el proceso de evaluación se definen 5 métricas: RMSE, MAE, Precisión, Recall y tiempo de entrenamiento. Para el entrenamiento y evaluación de los algoritmos se define un proceso de muestreo que selecciona los conjuntos de datos para cada etapa.
- Modelo de priorización de algoritmos: Este permite ordenar los algoritmos del sistema en orden descendente siendo mejor el primero de la lista, para ello se tienen en cuenta los resultados en las diferentes métricas de evaluación y los valores de importancia que defina el cliente específico para cada métrica. Al aplicar este modelo con los conjuntos de datos

evaluados, se encuentra que diferentes algoritmos presentan mejores resultados para diferentes conjuntos de datos. Sin embargo, los primeros lugares son ocupados por los mismos algoritmos en los tres casos de estudio. Además se realiza una prueba estadistica cuando hay poca diferencia entre los primeros, para garantizar la priorización de los algoritmos.

- Capa de servicios para consumo del sistema: los servicios básicos que permiten el consumo del sistema desde un punto de vista de configuración y ejecución de procesos, al igual que para el consumo de las recomendaciones desde los aplicativos de cada cliente. Esta capa de servicio no es implementada en el prototipo pero si se definen para una futura implementación.

TRABAJOS FUTUROS

El diseño del sistema de recomendación propuesto emplea múltiples procesos requeridos y opcionales para proveer recomendaciones a diferentes clientes. sin embargo, se reconocen varios puntos de mejora que pueden agregar mayor robustez al sistema y mejorar el rendimiento incurrido al momento de ejecutar cada uno de los procesos:

- Este sistema permite obtener los datos de diferentes clientes en diferentes formatos (CSV, JSON) y permite categorizarlos en diferentes fuentes de información. Los algoritmos del prototipo solo reciben datos explícitos para funcionar. Es recomendable adaptar otros algoritmos que hagan uso de otras fuentes de información como por ejemplo: características de usuarios, características de los artículos, interacciones implícitas, reglas de negocio, datos que acompañan la interacción, entre otros. De igual forma, al tener diferentes fuentes de información y algoritmos que las usen, se debe pensar

en un sistema de hibridación que permita hacer uso del mejor algoritmo seleccionado para cada fuente al momento de pedir una recomendación.

- Emplear otras técnicas de recomendación que permitan mejorar los resultados del proceso de hibridación y disponer de algoritmos que puedan ofrecer mejores resultados para las características de diferentes conjuntos de datos.
- Hacer uso de múltiples calificaciones para un mismo tipo de información. En algunos dominios se presentan múltiples aspectos de un artículo que pueden ser calificados independientemente. Para lograr un sistema más robusto, es necesario recibir estas entradas y adaptar algoritmos que las consuman y generen recomendaciones basadas en más de un aspecto.
- Algunos procesos del sistema podrían ser más óptimos computacionalmente si se aplican técnicas de computación paralela, pero esto dependerá mucho de la implementación en librerías de los principales lenguajes de uso, especialmente los procesamientos que se llevan a cabo durante la evaluación e hibridación de los algoritmos.
- Actualmente el sistema obtiene información como la fecha y hora en que se genera una interacción. Este tipo de información puede ser usada para separar los conjuntos de datos de entrenamiento y pruebas de una forma más sofisticada y que refleja mejor cómo lo hará cada algoritmo en producción.
- Implementar la capa de servicios y una interfaz web que permita a cada cliente configurar el sistema y ver los reportes y resultados que este entrega.
- Agregar otras métricas de evaluación de algoritmos como la evaluación del ranking de las recomendaciones.
- El proceso de priorización y selección del mejor algoritmo debería ejecutarse cada vez que las características de la población cambien con cierta significancia, esto se propone para investigaciones futuras, buscando que el sistema aprenda y detecte automáticamente cuando priorizar y seleccionar un nuevo algoritmo.

- Proponer un modelo que aprenda de los resultados en cada métrica y la priorización de los algoritmos, basandose en las características de los conjuntos de datos usados. Se requiere varios conjuntos de datos para entrenar este modelo.

REFERENCIAS

[1] X. Liu, “Context-Aware Recommender Systems for Implicit Data,” 2014.

[2] F. Ricci, L. Rokach, and B. Shapira, *Recommender Systems Handbook*. Boston, MA: Springer US, 2011.

[3] J. Leskovec, A. Rajaraman, and J. D. Ullman, “Recommendation Systems,” in *Mining of Massive Datasets*, Cambridge: Cambridge University Press, 2014, pp. 292–324.

[4] T.-T. Yang and H.-W. Tseng, “Numerical similarity-aware data partitioning for recommendations as a service,” in *Proceedings of the Symposium on Applied Computing - SAC ’17*, 2017, pp. 887–892.

[5] D. Ben-Shimon, A. Alexander Tsikinovsky, M. Friedmann, and J. Hörle, “Configuring and monitoring recommender system as a service,” in *Proceedings of the 8th ACM Conference on Recommender systems - RecSys ’14*, 2014, pp. 363–364.

[6] J. Paradinas, “EL CONSUMO Y LAS NECESIDADES HUMANAS.” [Online]. Available: https://www.acfilosofia.org/component/content/article?id=139:el-consumo-y-las-necesidades-humanas. [Accessed: 08-Nov-2017].

[7] Y. Yun, D. Hooshyar, J. Jo, and H. Lim, “Developing a hybrid collaborative filtering recommendation system with opinion mining on purchase review,” *J. Inf. Sci.*, p. 016555151769295, Feb. 2017.

[8] P. Lops, M. de Gemmis, and G. Semeraro, “Content-based Recommender Systems: State of the Art and Trends,” in *Recommender Systems Handbook*, Boston, MA: Springer US, 2011, pp. 73–105.

[9] Q. Liu and D. R. Karger, “Kibitz: End-to-End Recommendation System Builder,” *Proc. 9th ACM Conf. Recomm. Syst.*, pp. 335–336, 2015.

[10] “raccoon,” 2016. [Online]. Available: https://www.npmjs.com/package/raccoon. [Accessed: 02-Oct-2018].

[11] “easyrec :: open source recommendation engine,” 2016. [Online]. Available: http://easyrec.org/home. [Accessed: 02-Oct-2018].

[12] “Recombee API 2.1.0 documentation,” 2016. [Online]. Available: https://docs.recombee.com/index.html. [Accessed: 02-Oct-2018].

[13] J. Beel et al., “Introducing Mr. DLib,” in Proceeding of the 11th annual international ACM/IEEE joint conference on Digital libraries - JCDL ’11, 2011, p. 463.

[14] “SuggestGrid | Hosted recommendation and similarity as a service.” [Online]. Available: https://www.suggestgrid.com/#/0. [Accessed: 02-Oct-2018].

[15] D. Stern, R. Herbrich, and T. Graepel, “Matchbox: Large Scale Bayesian Recommendations.” 01-Jan-2009.

[16] A. Freno and Antonino, “Practical Lessons from Developing a Large-Scale Recommender System at Zalando,” in *Proceedings of the Eleventh ACM Conference on Recommender Systems - RecSys ’17*, 2017, pp. 251–259.

[17] R. Burke, “Hybrid Web Recommender Systems,” in *The Adaptive Web*, Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 377–408.

[18] K. Peffers, T. Tuunanen, M. A. Rothenberger, and S. Chatterjee, “A Design Science Research Methodology for Information Systems Research,” *J. Manag. Inf. Syst.*, vol. 24, no. 3, pp. 45–77, Dec. 2007.

[19] P. Resnick and H. R. Varian, “Recommender systems,” *Commun. ACM*, vol. 40, no. 3, pp. 56–58, Mar. 1997.

[20] K. Patel, A. Thakkar, C. Shah, and K. Makvana, “A State of Art Survey on Shilling Attack in Collaborative Filtering Based Recommendation System,” Springer, Cham, 2016, pp. 377–385.

[21] G. Shani and A. Gunawardana, “Evaluating Recommendation Systems,” in *Recommender Systems Handbook*, Boston, MA: Springer US, 2011, pp. 257–297.

[22] G. Suganeshwari and S. P. Syed Ibrahim, “A Survey on Collaborative Filtering Based Recommendation System,” Springer, Cham, 2016, pp. 503–518.

[23] S. Berkovsky, T. Kuflik, and F. Ricci, “Mediation of user models for enhanced personalization in recommender systems,” *User Model. User-adapt. Interact.*, vol. 18, no. 3, pp. 245–286, Aug. 2008.

[24] C. Desrosiers and G. Karypis, “A Comprehensive Survey of Neighborhood-based Recommendation Methods,” in *Recommender Systems Handbook*, Boston, MA: Springer US, 2011, pp. 107–144.

[25] H. Steck and Harald, “Item popularity and recommendation accuracy,” in *Proceedings of the fifth ACM conference on Recommender systems - RecSys ’11*, 2011, p. 125.

[26] G. Adomavicius and A. Tuzhilin, “Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions,” *IEEE Trans. Knowl. Data Eng.*, vol. 17, no. 6, pp. 734–749, Jun. 2005.

[27] J. Chakraborty and V. Verma, “A survey of diversification techniques in Recommendation Systems,” in *2016 International Conference on Data Mining and Advanced Computing (SAPIENCE)*, 2016, pp. 35–40.

[28] J. A. Konstan, A. Borchers, B. Sarwar, J. Herlocker, and J. Riedl, *Combining collaborative filtering with personal agents for better recommendations*. AAAI Press, 1999.

[29] A. Felfernig, G. Friedrich, D. Jannach, and M. Zanker, “Developing Constraint-based Recommenders,” in *Recommender Systems Handbook*, Boston, MA: Springer US, 2011, pp. 187–215.

[30] F. Lorenzi and F. Ricci, “Case-Based Recommender Systems: A Unifying View,” Springer, Berlin, Heidelberg, 2005, pp. 89–113.

[31] I. Guy *et al.*, “Personalized recommendation of social software items based on social relations,” in *Proceedings of the third ACM conference on Recommender systems - RecSys ’09*, 2009, p. 53.

[32] R. Burke, “Hybrid Recommender Systems: Survey and Experiments,” *User Model. User-adapt. Interact.*, vol. 12, no. 4, pp. 331–370, 2002.

[33] N. Hug, “{S}urprise, a {P}ython library for recommender systems.” 2017.

[34] “Factor in the Neighbors: Scalable and Accurate Collaborative Filtering YEHUDA KOREN Yahoo! Research.”

[35] G. Takács, I. Pilászy, B. Németh, and D. Tikk, “Matrix factorization and neighbor based algorithms for the netflix prize problem,” in *Proceedings of the 2008 ACM conference on Recommender systems - RecSys ’08*, 2008, p. 267.

[36] Y. Koren and R. Bell, “Advances in Collaborative Filtering,” in *Recommender Systems Handbook*, Boston, MA: Springer US, 2011, pp. 145–186.

[37] Xin Luo, Mengchu Zhou, Yunni Xia, and Qingsheng Zhu, “An Efficient Non-Negative Matrix-Factorization-Based Approach to Collaborative Filtering for Recommender Systems,” *IEEE Trans. Ind. Informatics*, vol. 10, no. 2, pp. 1273–1284, May 2014.

[38] P. Cremonesi, R. Turrin, E. Lentini, and M. Matteucci, “An evaluation Methodology for Collaborative Recommender Systems,” 2010.

[39] P. Vassiliadis, A. Simitsis, and S. Skiadopoulos, “Conceptual modeling for ETL processes,” in *Proceedings of the 5th ACM international workshop on Data Warehousing and OLAP - DOLAP ’02*, 2002, pp. 14–21.

[40] S. Sheth, N. Arora, C. Murphy, and G. Kaiser, “The weHelp reference architecture for community-driven recommender systems,” in *Proceedings of the*

2nd International Workshop on Recommendation Systems for Software Engineering - RSSE ’10, 2010, pp. 46–47.

[41] X. Amatriain and J. Basilico, “System Architectures for Personalization and Recommendation,” *Netflix Technology Blog*, 2013. [Online]. Available: https://medium.com/netflix-techblog/system-architectures-for-personalization-and-recommendation-e081aa94b5d8. [Accessed: 13-Jun-2018].

[42] B. Mohamed, B. Abdelkader, and B. F. M’hamed, “A multi-level approach for mobile recommendation of services,” in *Proceedings of the International Conference on Internet of things and Cloud Computing - ICC ’16*, 2016, pp. 1–6.

[43] F. Narducci, M. de Gemmis, and P. Lops, “A General Architecture for an Emotion-aware Content-based Recommender System,” in *Proceedings of the 3rd Workshop on Emotions and Personality in Personalized Systems 2015 - EMPIRE ’15*, 2015, pp. 3–6.

[44] P. B. Brazdil, C. Soares, and J. P. da Costa, “Ranking Learning Algorithms: Using IBL and Meta-Learning on Accuracy and Time Results,” *Mach. Learn.*, vol. 50, no. 3, pp. 251–277, 2003.

[45] G. Adomavicius and J. Zhang, “Impact of data characteristics on recommender systems performance,” *ACM Trans. Manag. Inf. Syst.*, vol. 3, no. 1, pp. 1–17, Apr. 2012.

[46] J. Griffith, C. O’Riordan, and H. Sorensen, “Investigations into user rating information and predictive accuracy in a collaborative filtering domain,” in *Proceedings of the 27th Annual ACM Symposium on Applied Computing - SAC ’12*, 2012, p. 937.

[47] P. Cremonesi, Y. Koren, and R. Turrin, “Performance of recommender algorithms on top-n recommendation tasks,” in *Proceedings of the fourth ACM conference on Recommender systems - RecSys ’10*, 2010, p. 39.

[48] G. Mustafa and I. Frommholz, “Performance comparison of top N recommendation algorithms,” in *2015 Fourth International Conference on Future Generation Communication Technology (FGCT)*, 2015, pp. 1–6.